HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

DIPLOMARBEIT

Herr
Anton Bors

Analyse und Implementierung eines nationalen Gateways

2016

Fakultät **Angewandte Computer- und Biowissenschaften**

DIPLOMARBEIT

Analyse und Implementierung eines nationalen Gateways

Autor:
Anton Bors

Studiengang:
Technische Informatik

Seminargruppe:
KT12wWA-F

Erstprüfer:
Prof. Dr.-Ing. Thomas Beierlein

Zweitprüfer:
Mag. Gernot Gebhart

Mittweida, September 2016

Bibliografische Angaben

Bors, Anton: Analyse und Implementierung eines nationalen Gateways, 121 Seiten, 34 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Diplomarbeit, 2016

Satz: LaTeX

Referat

Diese Arbeit beinhaltet die Thematik der Anforderungsanalyse mittels UML und die Umsetzung eine nationalen Gateways.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Quelltextverzeichnis

V. Abkürzungsverzeichnis

bzw. beziehungsweise

CDTA Character DATA
CRUD Create, Read, Update and Delete
CXF Apache CXF

d.h. das heißt
DAO Data Access Object
DEA Deterministisch endlicher Automat
DTO Data Transfer Object

EA EnterpriseArchitect
EESSI Electronic Exchange of Social Security Information
EJB EnterpriseJavaBean
EU Europäischen Union

FOP Apache FOP

IDE Integrierte Entwicklungsumgebung
IREB International Requirements Engineering Board

JAXB Java Architecture for XML Binding
JBoss AS JBoss 7 Application Server
JSR Java Specification Request

MTOM Message Transmission Optimization Mechanism

NEA Nichtdeterministisch endlicher Automat
NG Nationales Gateway

OMG Object Managing Group
Oracle Oracle Corporation
oä. oder ähnliches

PDF Portable Document Format

RE Requirements Engineering
RedHat Red Hat, Inc.

SH Stakeholder
SOAP Simple Object Access Protocol
SPoC Single Point of Contact
SysML System Modeling Language

u.dgl. und dergleichen

UC Use-Case
UML Unified Modeling Language
UUID Universally unique identifier

Vgl. Vergleiche
VT österreichischer Sozialversicherungsträger

WSDL Web Services Description Language

XML eXtensible Markup Language
XPath eXtensible Markup Language (XML) Path Language
XSD XML-Schema
XSL eXtensible Stylesheet Language
XSL-FO eXtensible Stylesheet Language - Formatting Objects

z.B. zum Beispiel

VI. Vorwort

Diese Arbeit widme ich meiner Frau und meinen Kindern, welche die Geduld und Muse hatten, mir die Zeit zu gönnen, dass ich diese Arbeit alleine und ohne Störungen schreiben konnte.
Des weiteren möchte ich meinem Arbeitskollegen Gernot Gebhart danken, dass er sich die Zeit nahm und mir als 2. Prüfer zur Verfügung stand und mir mit hilfreichen Tipps half, diese Arbeit in eine korrekte Form zu bringen.

1 Einleitung

Der Nachrichtentransfer zwischen den europäischen Staaten wächst stetig und ist mittels der Post einfach schon zu langsam geworden. Der Druck der Gesellschaft nach schnelleren Methoden und Möglichkeiten wird daher größer. Die Europäischen Union (EU) hat deshalb veranlasst, dass dieser Zustand zu ändern ist. Aus diesem Grund beschäftige ich mich in dieser Arbeit mit der Thematik einer Ankopplung an das derzeit noch nicht vorhandene europäische System, welches ich hiermit analysieren und implementieren werde.
Im wesentlichen werden hier die Methoden der Analyse mittels UML und die anschließende Implementierung des Systems beschrieben.
Das Hauptaugenmerk liegt dabei auf den innerstaatlichen Schnittstellen. Diese werden von mir entwickelt und erläutert. Des weiteren werde ich zeigen, wie diese auf einem Applikationsserver in Betrieb genommen werden können und wie man so eine Schnittstelle mit dem „Contract-First“ - Prinzip entwickeln kann und warum dieser Ansatz gewählt wurde.

2 Theoretische Grundlagen

Bevor man ein Softwareprojekt startet, sollten folgende Fragen geklärt werden::

- Was soll die Software machen ?
- Welchen Nutzen soll die Software bringen ?
- Wie kann der Projektplan dafür aussehen ?

Wenn diese Fragen sinnvoll beantwortet werden können, kann man damit beginnen, das Projekt auf zu setzen.

2.1 Was ist ein Projekt

Der SpringerVerlag gibt zum Thema Projekt folgende Definition:
Ein Projekt ist eine zeitlich befristete, relativ innovative und risikobehaftete Aufgabe von erheblicher Komplexität, die aufgrund ihrer Schwierigkeit und Bedeutung meist ein gesondertes Projektmanagement erfordert. [VS]
Eine weitere Definition auf Wikipedia, beschreibt ein Projekt aus der Sicht des Projektdreiecks:
Ein Projekt ist ein zielgerichtetes, einmaliges Vorhaben, das aus einem Satz von abgestimmten, gelenkten Tätigkeiten mit Anfangs- und Endtermin besteht und durchgeführt wird, um unter Berücksichtigung von Zwängen bezüglich Zeit, Ressourcen (zum Beispiel Geld bzw. Kosten, Produktions- und Arbeitsbedingungen, Personal) und Qualität ein Ziel zu erreichen. [WAc]
Nach den vorangegangen Definitionen kann man den Schluss ziehen, dass ein Projekt nicht einfach nur die Summe von Tätigkeiten zum Erreichen eines Zieles ist. Wie Abbildung 2.1 zeigt, besteht ein Zusammenhang aus Kosten, Ergebnis und der Dauer.
Dieses Dreieck ist ein Grund dafür, warum einige Softwareprojekte, und auch andere Projekte, mit Mehrkosten abgeschlossen wurden, oder gar nicht zum Abschluss kamen, oder diese zwar abgeschlossen wurden, jedoch einige Anforderungen nicht umgesetzt, beziehungsweise (bzw.) nur rudimentär implementiert wurden. Meist liegt der genauere Grund dafür darin, dass in der Analysephase der Projekte einigen Faktoren zu wenig Beachtung geschenkt, bzw. diese gar nicht erwähnt wurden. Manchmal werden diese Faktoren sogar betrachtet und als unwichtig erachtet. Um diesem Verhalten entgegen zu wirken, wurde die Disziplin des Requirements Engineering (RE) eingeführt.

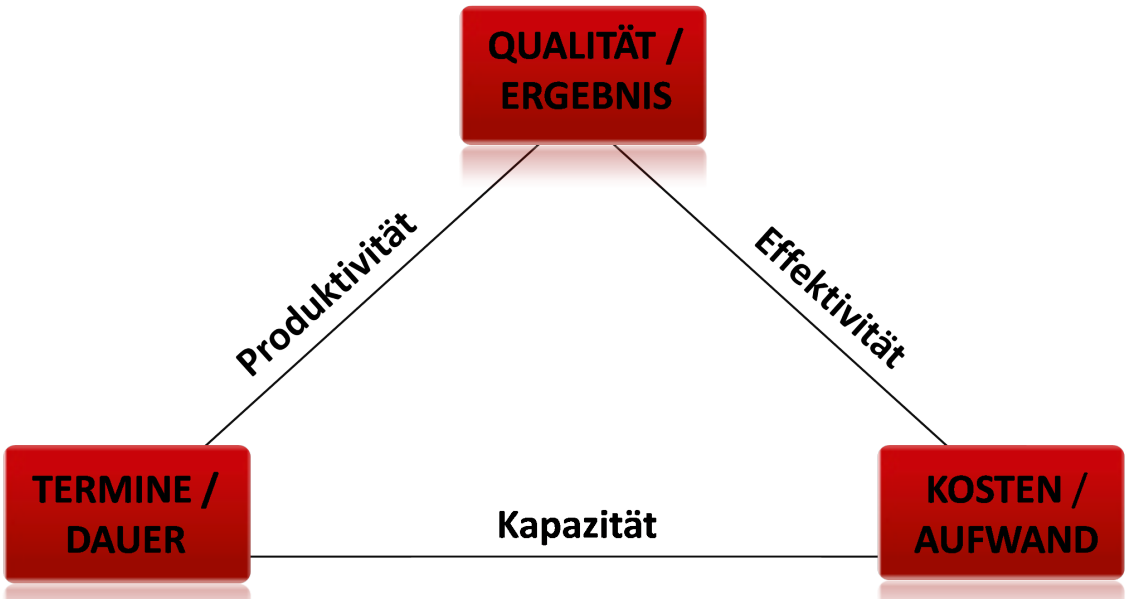

Abbildung 2.1: Magisches Dreieck des Projektmanagements

2.2 Was ist Requirements Engineering

Die Definition von RE ist der International Requirements Engineering Board (IREB) nach: [PR15, p. 4]

> Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation von Anforderungen mit folgenden Zielen:
>
> 1. Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen systematisch zu managen.
> 2. Die Wünsche und Bedürfnisse der Stakeholder (SH) zu verstehen, zu dokumentieren sowie die Anforderungen zu spezifizieren und zu managen, um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der SH entspricht.

Aus dieser Definition heraus ergibt sich auch schon der zeitliche Umfang des REs. Man muss mit dem RE starten, wenn das Projekt startet. Das Ende des RE ist nicht ganz so einfach zu finden, da wenn ein Projekt beendet wird, heißt dies nicht zwangsläufig, dass auch das RE beendet wird. Als Faustregel gilt, dass wenn ein Projekt als gescheitert gilt, das RE beendet wird. Sollte ein Projekt erfolgreich abgeschlossen und die Software in Wartung und Betrieb übernommen werden, so wird auch das RE weiter betrieben. Sollte es zu Änderungen der Software, oder ähnliches (oä.) kommen, so werden diese genauso dokumentiert und in die bestehenden Anforderungen aufgenommen und mit gewartet.

Damit ein Projekt innerhalb seiner Grenzen bleibt, müssen diese Grenzen auch bekannt sein. Zum einen gibt es die Grenzen aus Sicht des Projektmanagements, welche die zeitlichen, budgetären und qualitativen Grenzen bildet. Zum anderen gibt es die Systemgrenzen, welche das zu entwickelnde System in seinen Funktionen und Interaktionen eingrenzt.

2.2.1 Was ist ein System

Damit die Frage, was ein System ist, allgemeiner gehalten wird, zeigt die Abbildung 2.2 sinnbildlich, was man sich unter einem System und seinen Grenzen vorstellen kann.

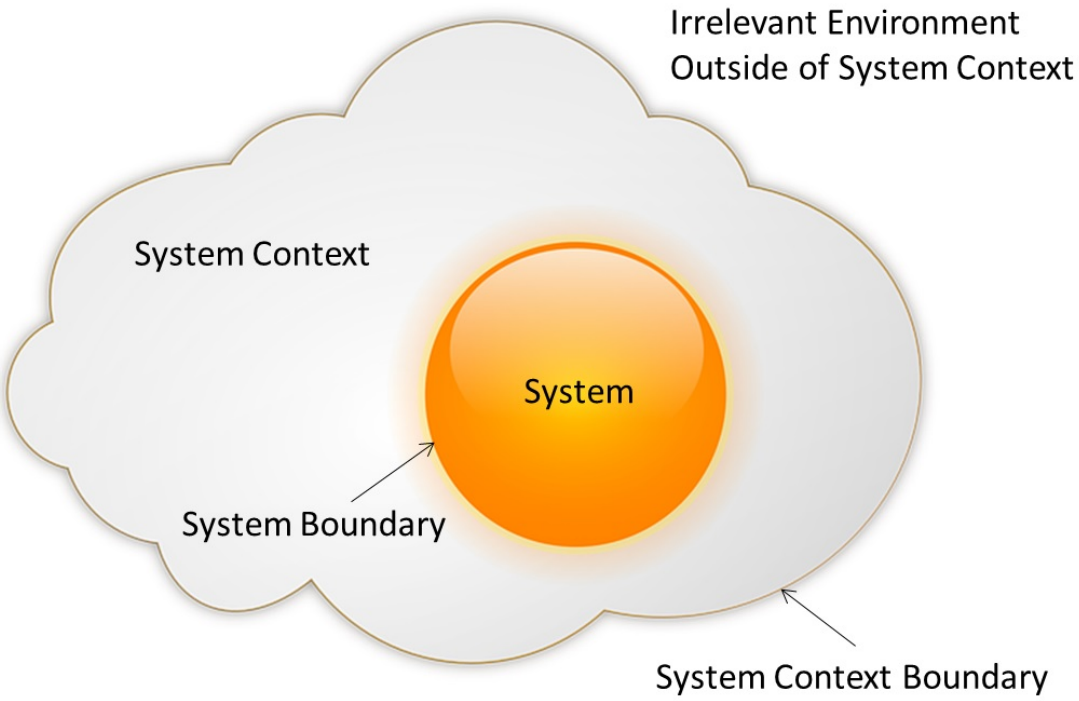

Abbildung 2.2: Der Systemkontext [Gmb]

Hierbei sieht man, dass das System selbst in einem Systemkontext eingebettet ist und dieser die Grenzen des Systems festlegt (Systemgrenze). Der Systemkontext wiederum wird von der Systemkontextgrenze umgeben, welche den Kontext von allem irrelevanten trennt. Was als irrelevant und was als relevant eingestuft wird, wird von den SH vorgegeben, da diese das System spezifizieren und vorgeben. Wenn diese Grenzen nicht bekannt sind, oder nicht gezogen werden, so besteht die Gefahr, dass am Ende das System etwas ganz anderes macht, als ursprünglich geplant, bzw. definiert war.

2.2.2 Die Anforderungen

Eine Anforderung stellt eine Eigenschaft des Systems dar. Die IREB gibt hier folgende Definition: [PR15, p. 3]

> Eine Anforderung ist:
>
> 1. Eine Bedingung oder Fähigkeit, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
> 2. Eine Bedingung oder Fähigkeit, die ein System oder Teilsystem erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
> 3. Eine dokumentiere Repräsentation einer Bedingung oder Eigenschaft gemäß 1 oder 2.

Daraus geht hervor, dass eine Anforderung immer direkt mit dem System zu tun haben muss, damit es auch eine Anforderung in diesem Systemkontext ist. Die Anforderung „Das Wetter soll schön sein“, ist keine Anforderung, welche eine Eigenschaft des Systems beschreibt. Die Anforderung „Der Anzeigehintergrund muss blau sein“ hingegen, stellt eine direkte Anforderung an das System dar.
Anforderungen durchlaufen üblicherweise einen Zyklus von mehreren Schritten, damit diese auch als Anforderungen aufgenommen oder auch abgelehnt werden. Wenn eine Anforderung abgelehnt wurde, so heißt dies nicht, dass man diese Anforderung vergessen sollte, sondern, dass diese mitgenommen wird und der Grund für die Ablehnung festgehalten wird, damit diese Anforderung, sollte diese nochmals vorkommen, nicht noch einmal den Zyklus durchlaufen muss[1].

2.2.3 Der Anforderungszyklus

Der Prozess des Findens von Anforderungen besteht im wesentlichen aus 4 Schritten. Abbildung 2.3 beschreibt den Prozess des Findens von Anforderungen. Es wird immer mit dem Schritt „Ermitteln einer Anforderung“ begonnen.

[1] Siehe Kapitel 2.2.3

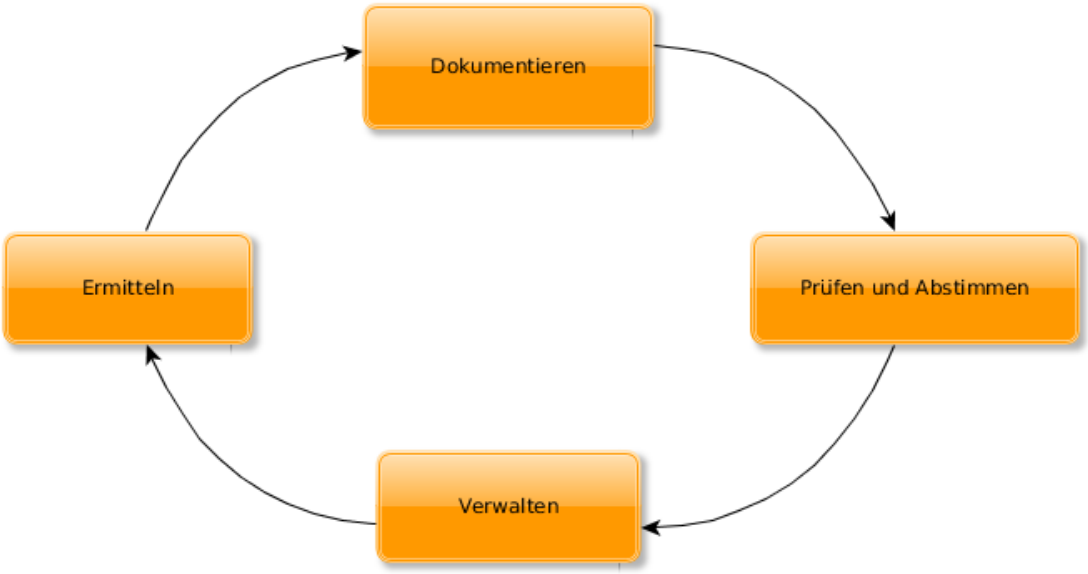

Abbildung 2.3: Der Anforderungszyklus

2.2.3.1 Anforderungen ermitteln

Anforderung wachsen leider nicht auf Bäumen. Man muss diese ermitteln. Anforderungen können aus folgenden Quellen ermittelt werden:

- SH
  Ein Stakeholder ist eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat [PR15, p. 4]
- Dokumente
  Alle Arten der schriftlichen Dokumentation können hier verwendet werden. Zum Beispiel (z.B.):
  - Gesetzestexte
  - Normen
  - Schnittstellenbeschreibungen
  - Anforderungsdokumente von Altsystemen
- Systeme in Betrieb
  Gibt es schon ein altes System, so sind in der Regel die Funktionalitäten des alten Systems in irgendeiner Form zu übernehmen. Deshalb eignen sich diese auch als Quelle, um Erfahrungen über die Abläufe, Strukturen und dergleichen (u.dgl.) zu sammeln, wobei hier wiederum die schriftliche Dokumentation im Vordergrund steht. Falls es diese nicht gibt, kann man auch LLearning by Doingßum Einsatz bringen und einem Keyuser[2] über die Schultern schauen.

[2] Ein Keyuser ist eine Person, oder Personengruppe, welche das abzulösende System gut kennen und die damit verbundenen Abläufe erklären können.

2.2.3.2 Anforderungen dokumentieren

Wenn man eine Anforderung ermittelt hat, dann muss diese auch dokumentiert werden. Es gibt 3 Arten der Dokumentation

1. natürliche Sprache
   Die natürliche Sprache als Form der Dokumentation hat ihre Vor- und Nachteile. Ein Vorteil ist, dass sie universell einsetzbar ist. Alle, die diese Sprache können, verstehen auch den Text und können diesen interpretieren und auslegen. Somit kann man etwas beschreiben und anderen vermitteln, um was es geht. Ein Nachteil der natürlichen Sprache ist die Interpretation und Auslegung dieser. Ein immer wiederkehrendes Beispiel dafür ist „Das System muss die Anfrage schnell verarbeiten.“ Für einen Geologen wäre es auch noch schnell, wenn die Dauer 10 Jahre betragen würde, hingegen für einen Blitzforscher eine Dauer von 2 Minuten schon sehr lange ist.
2. modellbasierte Dokumentation
   Ein Modell ist ein abstraktes Abbild der Wirklichkeit oder einer möglichen Wirklichkeit. Das Modell ist für die Dokumentation nicht universell einsetzbar und besitzt deshalb dafür definierte Diagramme, welche einen Sachverhalt beschreiben sollen. Der Vorteil von Diagrammen ist, dass man mit wenig Aufwand schnell etwas bildlich beschreiben kann und jeder, der die Notation kennt, das Diagramm gleich interpretiert. Der Nachteil dabei ist, dass man die Notationen kennen muss, um damit sinnvoll arbeiten zu können. Die verwendeten Diagramme werden noch im Kapitel 2.3 genauer behandelt.
3. Mischform aus 1 und 2
   Damit sind alle Arten von Bildern, welche mit Text zum besseren Verständnis versehen sind, gemeint, genauso wie Text, welcher mit Bildern veranschaulicht wird.

2.2.3.3 Anforderungen prüfen und abstimmen

Das Prüfen und Abstimmen von Anforderungen hat den Zweck, Anforderungen gegen gewisse Qualitätsstandards zu prüfen und diese mit den Quellen abzustimmen, ob die Anforderung auch für jeden, der mit dem Projekt betraut ist und mitwirkt, das gleiche bedeutet und nicht widersprüchlich zu anderen Anforderungen ist. Dieser Vorgang ist aber nicht nur auf einzelne Anforderung anzuwenden, sondern sollte auch auf alle entstandenen Dokumentationen angewandt werden.

2.2.3.4 Anforderungen verwalten

Das Verwalten von Anforderungen ist der am längsten andauernde Prozess des Anforderungsmanagements. Eine Anforderung durchläuft, über die gesamte Dauer des

Projektes und darüber hinaus, mehrere Status. Das kann beginnen bei „neu“ und kann bei „umgesetzt“ oder „veraltet“ , „abgelehnt“ enden. Wichtig dabei ist, dass alle Anforderung zu verwalten sind, auch jene die abgelehnt wurden und somit keinen Bezug zum System haben sollten. Da jedoch irgendjemand befunden hat, dass diese Anforderung wichtig sein könnte, muss diese mit verwaltet werden.

2.2.4 Anforderungsdiagramm

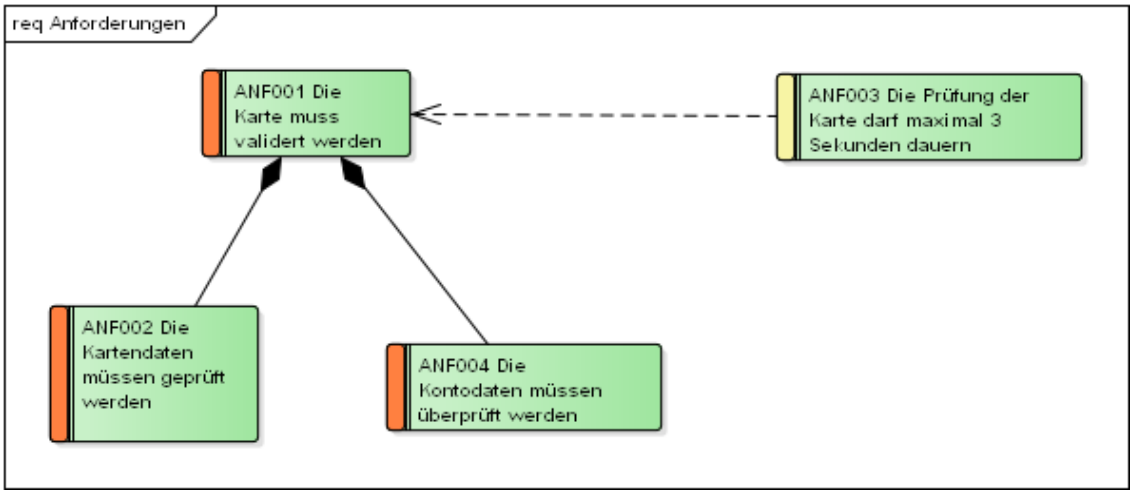

Abbildung 2.4: Anforderungsdiagramm

Das Anforderungsdiagramm ist kein Diagramm nach dem UML-Standard[3], sondern soll veranschaulichen, wie einzelne Anforderungen zusammenhängen und welche es gibt. Da die Nummerierung der Anforderungen nicht verändert werden darf, kann man zwar im Diagramm abgelehnte Anforderungen löschen, wobei es besser ist, dass diese einen Status gelöscht oder obsolet bekommen und somit ersichtlich wird, dass diese Anforderungen nicht zu beachten sind und so auch nicht mehr in den Prozess aufgenommen werden können.
In Abbildung 2.4 wird ein kleiner Auszug an Anforderungen abgebildet. Darin kann man sehen, dass sich die Validierung der Karte aus den Anforderungen „Die Kartendaten müssen geprüft werden“ und „Die Kontodaten müssen überprüft werden“ zusammensetzt. Aus den Anforderungen kann man so schon einen Einblick in den Ablauf und die geforderte Architektur der Software erhalten und kann diese Anforderungen innerhalb des Testzyklus und der Qualitätssicherung überprüfen.

[3] Vergleiche (Vgl.) Kapitel 2.3

2.3 Was ist UML

Unified Modeling Language (UML) ist eine Form der Dokumentation, welche aus der heutigen Sicht in der Softwareentwicklung mittels objektorientierten Sprachen nicht mehr weg zu denken ist. Vor allem bei größeren Projekten ist diese Form der Dokumentation für komplexere Abläufe und Strukturen eine standardisierte Dokumentationsform. Wie man in Abbildung 2.5 sieht, teilen sich die Diagramme in 2 Gruppen auf:

- Strukturdiagramme
- Verhaltensdiagramme

Im nachfolgenden wird auf jedes dieser Diagramme eingegangen, wobei die Diagramme, welche in der nachfolgenden Arbeit verwendet werden, genauer beschrieben werden. Um die einzelnen Diagramme besser veranschaulichen zu können, wird hier ein fiktives Projekt „Bankomat“ als Beispiel herangezogen, wobei die Komponenten willkürlich gewählt werden

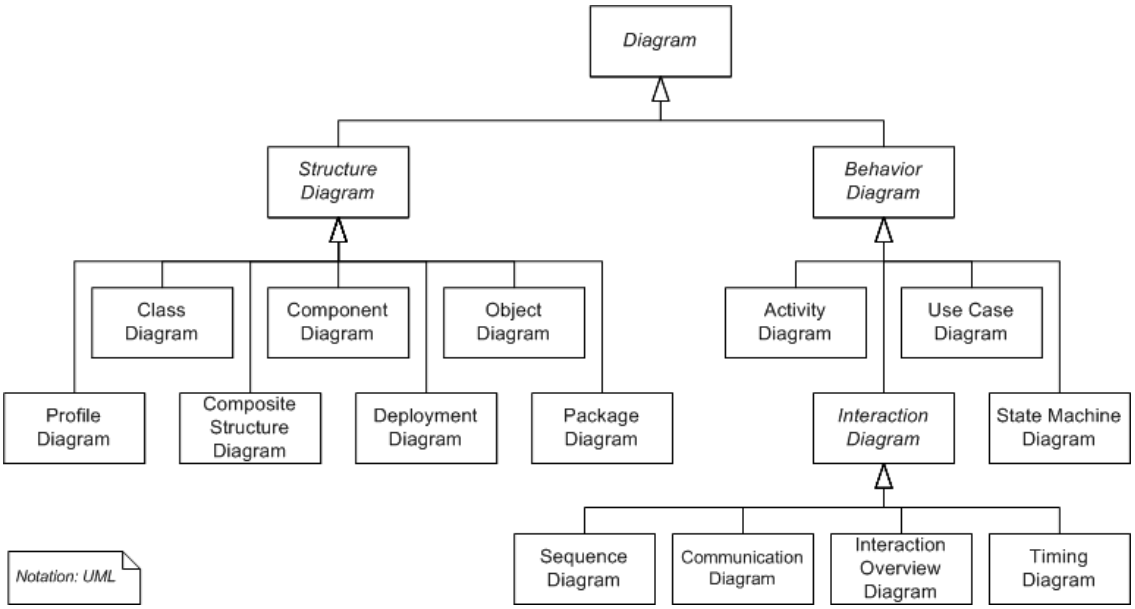

Abbildung 2.5: Die Diagramme der UML [WAa]

2.3.1 Strukturdiagramme

Als Strukturdiagramme werden alle jene Diagrammtypen bezeichnet, welche eine statische Komponente eines Systems darstellen. Die Instanzen dieser Komponenten können sich zwar ändern, aber die Struktur, das heißt (d.h.) die Komponente und die Beziehungen zu anderen Komponenten, bleibt über den gesamten Lebenszyklus konstant.

2.3.1.1 Klassendiagramm

Das Klassendiagramm stellt die Zusammenhänge der Klassen[4] untereinander dar.

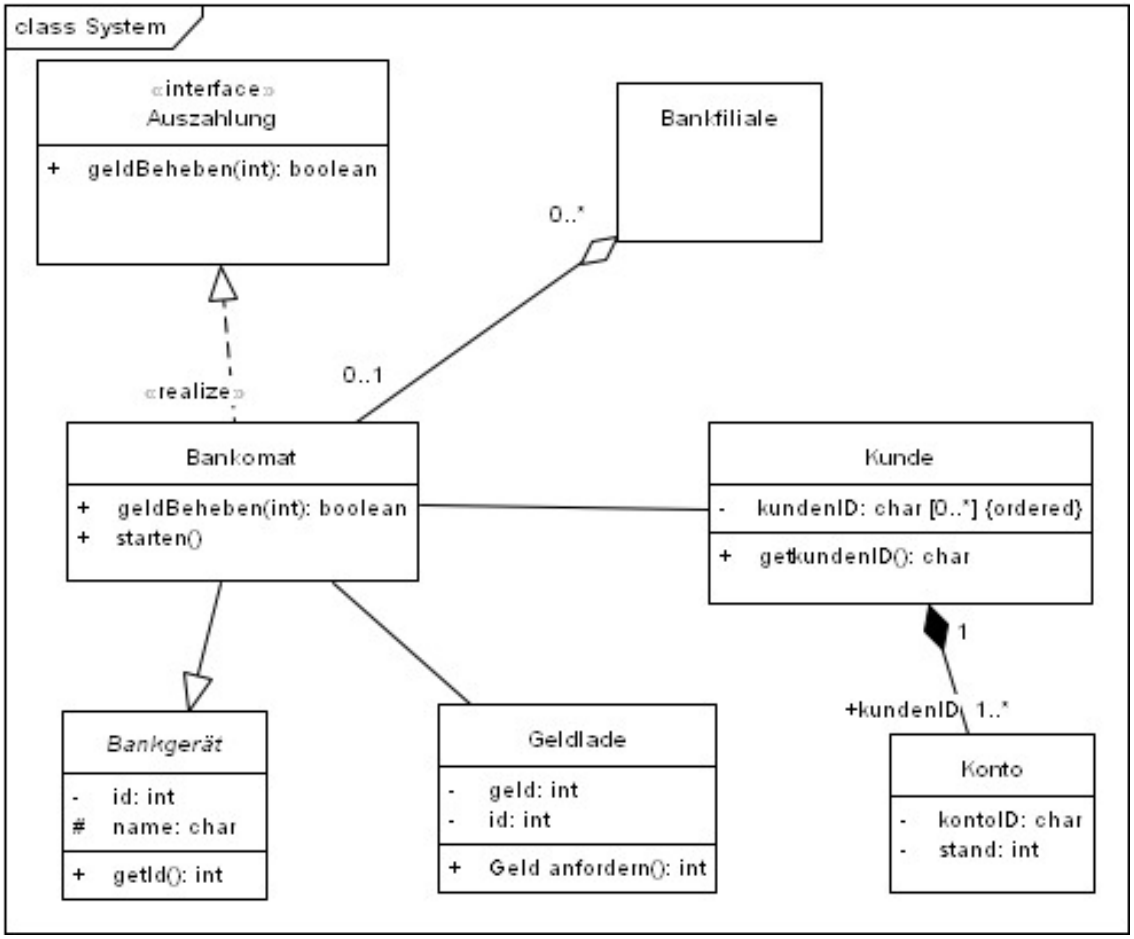

Abbildung 2.6: Das Klassendiagramm

In Abbildung 2.6 werden die einzelnen Möglichkeiten dargestellt.

2.3.1.1.1 Klassen Eine Klasse wird als Rechteck dargestellt, welches die Grenzen bildet. Darin muss es einen Namen geben, welcher die Klasse benennt. Oberhalb des Namens steht die Art der Klasse z.B. «interface», welche die Klasse näher spezifiziert. Bei normalen Klassen wird der Begriff «class» einfach weggelassen. Ist der Name der Klasse kursiv geschrieben, so bedeutet dies, dass dieses Klasse abstrakt ist und davon keine Instanz[5] gebildet werden kann.
Zur Bereichstrennung werden horizontale Linien benutzt. Die nächsten Bereiche sind

[4] Eine Klasse ist in der Softwareentwicklung ein abstrakter Begriff, für die Struktur eines Elements.
[5] Eine Instanz ist ein zur Laufzeit erzeugtes Objekt, welches nach der Beschreibung der Klasse erzeugt wird.

der Bereich der Attribute und der Bereich der Operationen.[6]

$$Attribut := [Sichtbarkeit]Name[:Typ][Multiplizität][=Vorgabewert][\{Eigenschaftswert\}] \tag{2.1}$$

$$Operation := [Sichtbarkeit]Name([\{Parameter\}])[:Rückgabetyp][\{Eigenschaftswert\}] \tag{2.2}$$

Wie man in den Deklarationen 2.1 und 2.2 sehen kann, ist der Name verpflichtend anzugeben und die Unterscheidung zwischen Attributen und Operationen stellen die Klammern nach dem Operationsnamen dar.
Die Sichtbarkeit in den beiden Deklarationen kann folgende Werte annehmen:

- + public jeder hat Zugriff
- - private nur die Klasse selbst hat Zugriff
- # protected alle abgeleiteten Klassen haben Zugriff
- $\sim$ package (kann auch weggelassen werden) alle Klassen im selben Namensraum haben Zugriff (Diese Option muss allerdings vom Compiler unterstützt werden)

Der Typ in der Deklaration 2.1 stellt den Datentyp des Attributes dar. Die Multiplizität gibt an, ob ein Attribut optional (0..1), verpflichtend(1[7]) oder mehrfach vorkommen kann. Bei einer Multiplizität > 1 werden in der Regel Listen oder Arrays verwendet. Der Vorgabewert stellt den Standardwert dar, welcher verwendet wird, wenn kein Wert vorher gesetzt wurde.Der Eigenschaftswert legt zusätzliche Eigenschaften fest und kann z.B. isReadOnly oder isOrdered annehmen.
Der Parameter stellt den Datentyp dar, welcher an die Operation übergeben werden muss. Der Name des Parameters spielt dabei keine Rolle, da die Reihenfolge der Parameter die Signatur[8] der Operation mitbestimmen. Der Rückgabetyp gibt den Datentyp an, welcher von der Operation geliefert wird. Erfolgt keine Rückgabe, so kann man auch das Wort „void“ verwenden.

2.3.1.1.2 Assoziationen Es gibt eine Vielzahl von möglichen Assoziationen. Die wichtigsten und häufigsten werden hier kurz vorgestellt:

- Generalisierung
  Die Klasse Bankomat ist auch ein Objekt vom Typ Bankgerät und erbt deshalb auch alle Elemente von Bankgerät. Wichtig ist hierbei, dass die Klasse Bankomat nicht auf das Attribut *id* der Klasse Bankgerät zugreifen kann, da die Sichtbarkeit *private* ist.
- Realisierung

[6] Notationshilfe: [] = Inhalt ist optional, {} = Inhalt kann öfter vorkommen und wird mittels Beistrich getrennt
[7] kann weggelassen werden
[8] Vergleiche Formel 2.2, wobei nur der Name und die Parameterliste ausschlaggebend sind.

Die Klasse Bankomat realisiert (implementiert) die Schnittstelle Auszahlung. Dies bedeutet, dass alle Operationen der Schnittstelle von Bankomat zu realisieren sind.

- Assoziation (einfache Linie)
  Die einfache Assoziation beschreibt, dass eine Klasse eine Beziehung zu einer anderen Klasse hat. Diese gibt aber noch keine Auskunft darüber, wie die Assoziation realisiert wird (Weiß die Klasse Kunde von der Klasse Bankomat, oder umgekehrt.)
- Komposition (Assoziation mit ausgefüllter Raute)
  Ein Objekt der Klasse Kunde kann nur existieren, wenn dieses auch mindestens ein Objekt der Klasse Konto besitzt.
- Aggregation (Assoziation mit leerer Raute)
  Eine Bankfiliale besitzt keinen, einen oder mehrere Bankomaten. Ein Bankomat gehört zu einer Bankfiliale. Das bedeutet, dass das Objekt der Bankfiliale auch existiert, wenn diese keinen Bankomat besitzt. Der Bankomat jedoch eine Bankfiliale benötigt um existent zu sein.

2.3.1.2 Kompositionsstrukturdiagramm

Das Kompositionsstrukturdiagramm zeigt die innere Struktur eines Elements[9] und die Interaktion mit seiner Umwelt, weshalb dieses Diagramm auch Architektur- oder Montagediagramm genannt wird.

2.3.1.3 Komponentendiagramm

Das Komponentendiagramm zeigt die Verteilung der Komponenten und wie diese miteinander kommunizieren. Hierbei steht nicht der Inhalt der Komponente im Vordergrund, sondern wie die Komponenten aufgebaut sind und über welche Kanäle kommuniziert wird. Das Symbol der Komponenten ist ein Rechteck mit abgerundeten Ecken, welches in der rechten oberen Ecke ein kleines Rechteck mit zwei weiteren Rechtecken besitzt.

2.3.1.4 Verteilungsdiagramm

Das Verteilungs- oder Deploymentdiagramm zeigt, welche Artefakte wohin ausgeliefert werden. Sollte die Software nicht auf einem Standort (Server, Webserver, Applikationsserver,..) laufen, so gibt dieses Diagramm Auskunft darüber, wo man ein Artefakt finden kann.

[9] Ein Element kann hierbei ein Artefakt, eine Objektklasse oder auch einfach die einzelnen Komponenten der Hardware sein.

2.3.1.5 Objektdiagramm

Das Objektdiagramm ist dem Klassendiagramm[10] sehr ähnlich, wobei das Objektmodell den inneren Zustand der Objekte zu einem Zeitpunkt zeigt. Da dieses Modell vom Klassendiagramm abgeleitet werden kann, können hier auch die Attribute mit Werten befüllt werden und zeigen so einen zeitlichen Zustand. Damit das Diagramm nicht überladen wird, werden hierbei nur jene Attribute gewählt, die gerade von Interesse sind.

2.3.1.6 Paketdiagramm

Das Paketdiagramm ist ein sehr universell verwendbares Diagramm. Es lässt sich nicht nur für softwareeigene Elemente (Packages in Java) verwenden, sondern auch um Informationen darüber zu geben, was in einem Auslieferungspaket enthalten ist, oder auch welche Elemente thematisch zusammen gehören.

2.3.1.7 Profildiagramm

Das Profildiagramm, auch nur Unified Modeling Language (UML)-Profil genannt, ist eine generische Erweiterung der UML, um auf spezielle Eigenheiten von anderen Systemen besser eingehen zu können.
Da UML für objektorientierte Softwaresprachen geschaffen wurde, fehlten in den ersten Version von UML die Möglichkeiten, diverse Aspekte von anderen Softwareprojekten abzubilden. Mit Hilfe von UML-Profilen ist dies nun möglich. Hierbei sei erwähnt, dass System Modeling Language (SysML) von der Object Managing Group (OMG) im Jahr 2003 als Modellierungssprache aufgenommen wurde und als UML-Profil nun abgebildet wird.

2.3.2 Verhaltensdiagramme

Verhaltensdiagramme modellieren den zeitlichen Verlauf der Komponenten eines Systems und deren Veränderungen. Daraus ergibt sich, dass diese Diagramme zu einem Zeitpunkt X einen Zustand haben können und zum Zeitpunkt Y einen anderen.

2.3.2.1 Aktivitätsdiagramm

Das Aktivitätsdiagramm stellt den Ablauf der Aktionen dar und gibt so einen Überblick, was getan werden muss. Jede Abweichung vom Ablauf stellt einen nicht definierten Zustand dar. Damit das Diagramm auch bei größeren Projekten übersichtlich bleibt, gibt es graphische Möglichkeiten, um das Diagramm nicht zu überladen.

[10] siehe 2.3.1.1

In Abbildung 2.7 sieht man einen vereinfachten Ablauf, welcher den UC „Geld abheben“ beschreibt. Eine wichtige Regel dabei ist, dass jedes Element einen Eingang und einen Ausgang hat. Die Ausnahmen dazu bilden *Fork / Join* und *Decision / Merge*. Der Startpunkt ist ein schwarzer, ausgefüllter Kreis, der nur einen Ausgangspfeil besitzt. Pro Aktivität darf nur ein Startpunkt vorhanden sein. Sind mehrere Startpunkte vorhanden, deutet dies auf Fehler in der Modellierung hin. Das Gegenstück dazu ist der Endzustand, welcher aus einen Kreis mit einem Punkt besteht und mehrfach vorhanden sein kann, da es auch mehrere definierte Endzustände gegeben kann. Die Aktivität ist gekennzeichnet als Rechteck mit runden Ecken und beinhaltet den Namen der Aktivität. Wenn eine Aktivität noch in kleinere Aktivitäten unterteilt werden kann, so kennzeichnet man diese Aktivität mit einem Zusatzsymbol, welches je nach Hersteller der Modellierungssoftware unterschiedlich ist. Die Notation sieht folgendes Symbol ⫛ vor, welches an einen kleinen Rechen erinnert, welcher darauf hinweist, dass diese Aktivität noch interne Schritte vorsieht. Im EnterpriseArchitect (EA) wird dieses Symbol mittels ∞ dargestellt.
Da Prozesse nie linear ablaufen, gibt es die Elemente *Decision* (Entscheidung) und *Merge* (Vereinigung), welche als Raute dargestellt werden. Die *Decision* gibt die Möglichkeit bedingte Verzweigungen zu machen z.B. „Pin OK?“. Die *Decision* hat immer einen Eingang und 2 Ausgänge, da man ja sonst in der Entscheidung stehen bleiben würde. *Merge* ist der Gegenpart zu *Decision*, benötigt immer zwei Eingänge und hat einen Ausgang. In der Regel sollte immer eine *Decision* zusammenführbar sein mit einem *Merge*. Des weiteren ist bei *Merge* wichtig, dass hier nur eine Eingabe notwendig ist, um zum nachfolgenden Punkt zu kommen.
Um gleichzeitige Aktivitäten zu signalisieren, gibt es das Element *Fork*, welches, genauso wie *Join* ein dicker schwarzer Strich ist. *Fork* kennzeichnet den gleichzeitigen Beginn von mehreren Aktivitäten und *Join* das Ende derer. Hierbei ist wichtig, dass *Join* immer auf alle Ergebnisse wartet und somit ein Synchronisierungspunkt ist. Wenn nun bei einem *Fork* 2 Aktivitäten gestartet werden, so müssen auch 2 Kontrollflüsse in den *Join* kommen. Es ist nicht erlaubt, dass weniger oder mehr zusammenkommen. Sollte man auf die Idee kommen, dass man nach einem *Fork* nochmals einen *Fork* benötigt, so muss dieser zuerst gesammelt werden und kann erst dann in letzten *Join* übergehen. Zur besseren Veranschaulichung werden in der Abbildung 2.7 die Aktivitäten „Karte entnehmen“ und „Geld abzählen“ parallel ausgeführt.

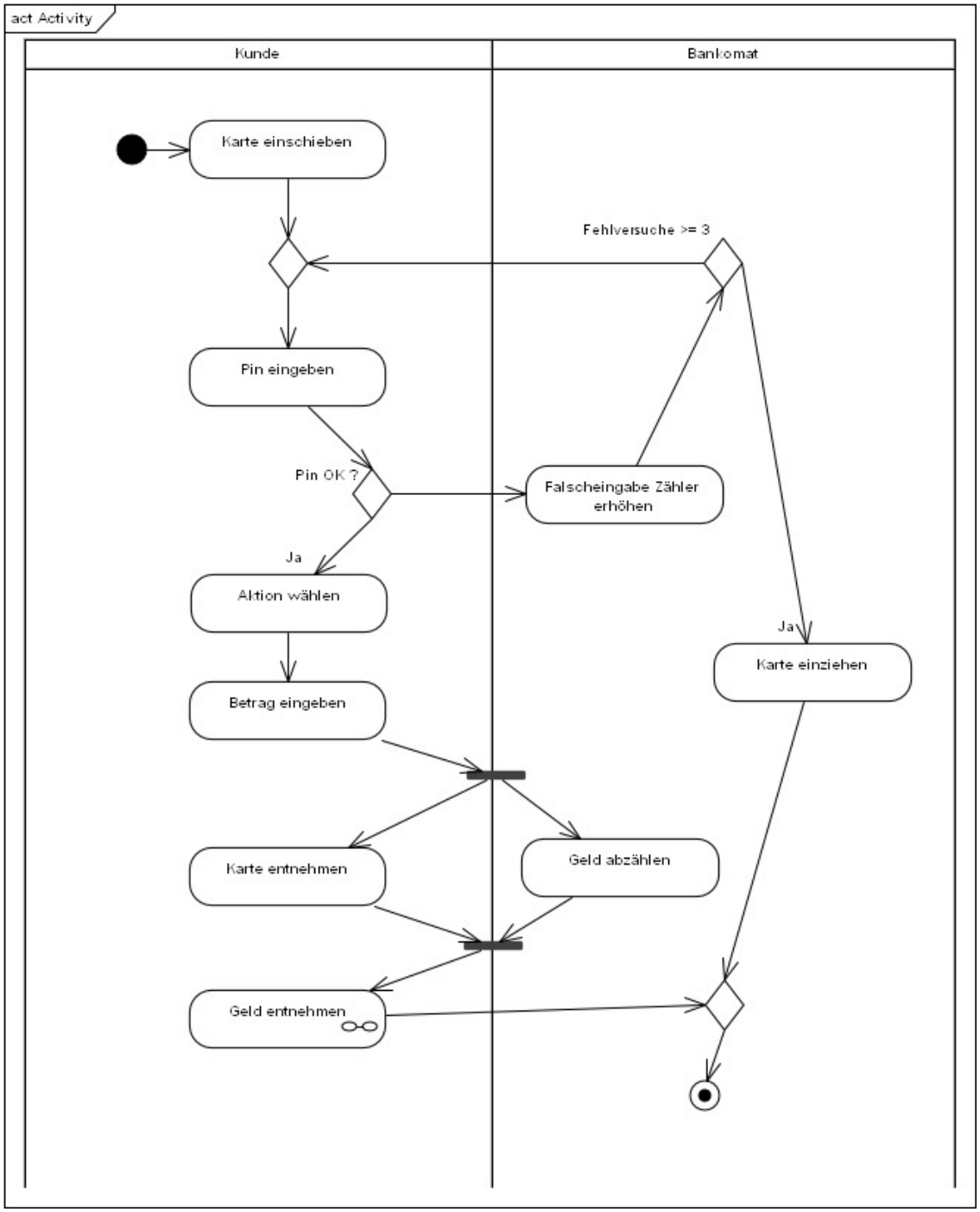

Abbildung 2.7: Aktivität: Geld abheben

2.3.2.2 Anwendungsfalldiagramm

Das Anwendungsfalldiagramm, auch UC dient der Darstellung der Interaktion der Akteure mit dem System und was diese tun können, bzw. was diese tun sollten.

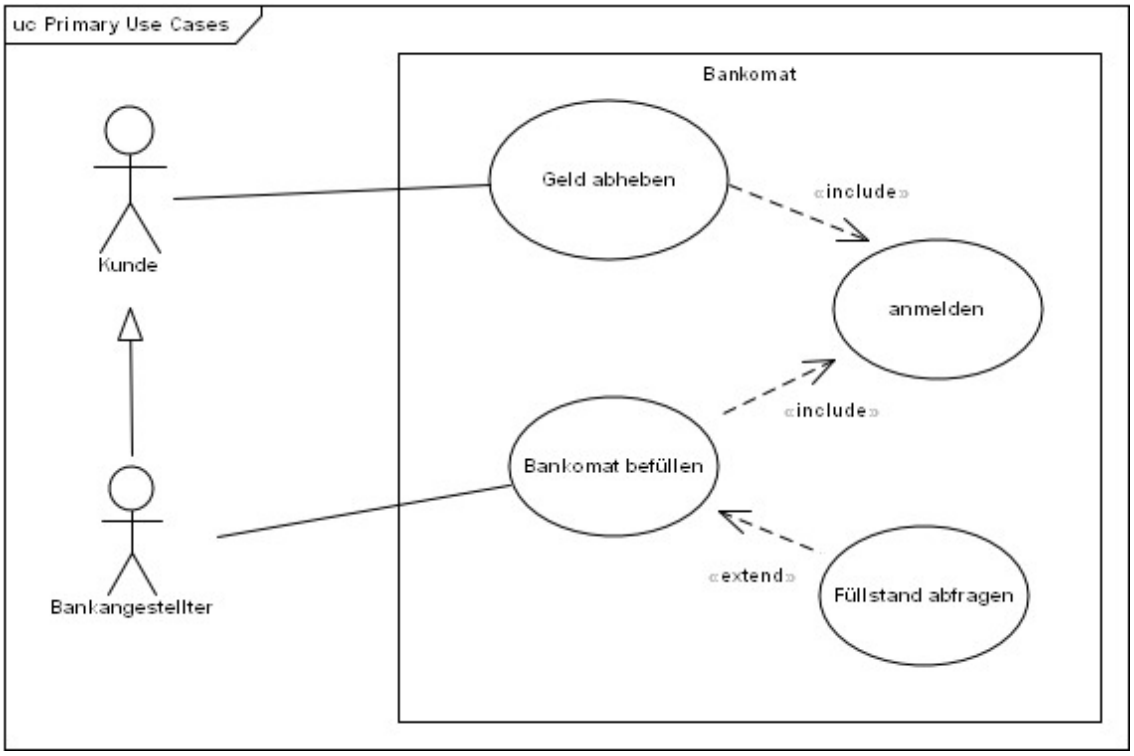

Abbildung 2.8: UC Diagramm

In Abbildung 2.8 sieht man das Zusammenspiel der einzelnen Komponenten. Der Rahmen des Systems bildet dabei die Systemgrenze. Alles, was innerhalb dieses Rahmens ist, gehört auch zum System dazu. Die Akteure stehen außerhalb des Systems und benutzen die Anwendungsfälle. Der Pfeil mit der weißen Spitze ist die Generalisierung und kommt aus der Vererbung. Das bedeutet, dass der Bankangestellte auch ein Kunde ist und alles was der Kunde kann auch der Bankangestellte. Der Pfeil mit der Beschriftung „«include»“ bedeutet, dass der UC, auf den gezeigt wird, bereits enthalten ist und auch verwendet werden muss. Der Pfeil mit der Beschriftung „«extend»“ bedeutet, dass der UC, auf den gezeigt wird, den UC von dem ausgegangen wird, verwenden kann.
Wichtig ist bei diesem Diagramm, dass der zeitliche Ablauf der Anwendungsfälle hier keine Rolle spielt und auch nicht mit modelliert wird. In unserem Beispiel könnte man Geld beheben, bevor man sich anmeldet. Des weiteren, gibt eine Assoziation (gerader Stich ohne Pfeil) auch keine Auskunft wer der Auslöser für die Verwendung ist. Es kann hier auch sein, dass der UC „Bankomat befüllen“ den Bankangestellten benutzt.

2.3.2.3 Interaktionsübersichtsdiagramm

Das Interaktionsübersichtsdiagramm ist eine Mischung aus dem Aktivitätsdiagramm[11] und dem Sequenzdiagramm[12]. Der Grundaufbau dieses Diagramms kommt aus dem

[11] Siehe dazu 2.3.2.1
[12] Siehe dazu 2.3.2.4

Aktivitätsdiagramm. Der Inhalt der Elemente kommt aus dem Sequenzdiagramm. Der Vorteil des Interaktionsübersichtsdiagramms ist, dass der generelle Ablauf der Aktionen ersichtlich wird und die dahinter liegenden Sequenzen referenziert, bzw. diese auch direkt innerhalb der Elemente modelliert werden können. Dies nimmt den Sequenzdiagrammen die Komplexität des gesamten und man bekommt den Überblick, wie die einzelnen Abläufe zusammenhängen. Des weiteren kann man somit auf eine Sequenz öfters referenzieren und muss diese nicht duplizieren.

2.3.2.4 Sequenzdiagramm

Das Sequenzdiagramm zeigt den Lebenslauf von Objekten und deren Nachrichtenaustausch. Dies bedeutet, dass man hieraus schon ablesen kann, zu welchem Zeitpunkt eine Instanz eines Objektes existiert, bzw. wann eine neue Instanz erzeugt wird.

2.3.2.5 Kommunikationsdiagramm

Das Kommunikationsdiagramm ist ein kompaktes Diagramm, welches dem Sequenzdiagramm gleicht. Hierbei steht die Kommunikation zwischen den Objekten im Vordergrund. Dabei ist das Hauptaugenmerk auf die Reihenfolge der Nachrichten, welche durch eine Nummerierung gekennzeichnet wird, und welche Nachrichten mit welchen Parametern gesendet werden. Hierbei gilt, dass die Nachricht mir der Nummer 1.2 vor der Nachricht 2 kommt. Die Nummerierung bildet hierbei nicht nur die logische Abfolge ab, sondern gruppiert die Nachrichten auch. Der Kontext der Gruppierung bleibt dem Modellierer überlassen. Man kann die Nachrichten z.B. nach der Startseite oder auch nach der Startaktion gruppieren.

2.3.2.6 Zeitverlaufsdiagramm

Das Zeitverlaufsdiagramm ist ein Diagramm, welches aus der Digitaltechnik stammt und den Zustand von Elementen zu einem Zeitpunkt zeigt. Dieses Diagramm ist dem Sequenzdiagramm[13] sehr ähnlich, wobei die Achsen um 90° gedreht sind und die Zeit von links nach rechts zunimmt.

2.3.2.7 Zustandsdiagramm

Das Zustandsdiagramm oder Zustandsautomat beschreibt die möglichen Zustände, welche innerhalb eines Ablaufs eintreten können und wie der Wechsel von einem Zustand in den anderen passiert. Dabei gibt es zwei Arten der Automaten:

[13] Siehe dazu 2.3.2.4

- Deterministisch endlicher Automat (DEA)
  Ein Zustandsautomat, welcher zu jedem Zeitpunkt weiß, wie man von einem Zustand in den nächsten kommt.
- Nichtdeterministisch endlicher Automat (NEA)
  Ein Zustandsautomat, bei dem es mindestens einen Zustand gibt, bei dem der Übergang auf einen der weiteren Folgezustände nicht entscheidbar ist.

Software sollte immer so geschrieben werden, dass der dahinter liegende Zustandsautomat deterministisch ist, da sonst ein Fehlverhalten und damit der Absturz der Laufzeitumgebung die Folge ist.

3 Aufgabenstellung

Mit der EU-Verordnung Nr. 883/2004 [Eurb] und der daraus resultierenden Durchführungsverordnung (EG) Nr. 987/2009 [Eura] ist jeder Staat der EU verpflichtet, den Datenaustausch für Systeme der sozialen Sicherheit auf ein elektronisches System, genannt Electronic Exchange of Social Security Information (EESSI), umzustellen. Der Vorteil dahinter ist der vereinheitlichte und schnellere Austausch von Daten innerhalb der EU.
In diesem Zuge muss auch Österreich sich an dieses System anbinden. Dazu soll ein nationales Gateway konstruiert und implementiert werden, welches diesen Datenaustausch gewährleisten soll. Da zum jetzigen Zeitpunkt (01.06.2016) die Vorgaben der Europäischen Kommission noch in der Abstimmungsphase sind, sollen hier nur die nationalen Elemente beleuchtet werden.
Damit die Anbindung Österreichs an das EESSI nahtlos erfolgen kann und die Umstellung für die österreichischen Versicherungsträger nur einmalige Kosten verursachen soll, wird ein Single Point of Contact (SPoC) eingerichtet, welcher den möglichen mehrmaligen Aufwand abfedern und so Stabilität in den innerösterreichischen Informationsaustausch bringen soll.
Es gibt innerhalb Österreichs ein definiertes Format, mit dem die Daten an die EU übermittelt werden sollen. Dieses nationale Format muss am nationalen Gateway in das EU-Format konvertiert werden und im Anschluss dem EU-System übergeben werden.
Damit der derzeitige Betrieb schon auf einen teilweise elektronischen Austausch umgestellt werden kann, wird die Ausgabe in Richtung EU derzeit als Portable Document Format (PDF) - Ausgabe implementiert.

3.1 Innerösterreichische Schnittstellen

Das Gateway soll über mehrere Simple Object Access Protocol (SOAP)-Schnittstellen verfügen, welche in folgende Kategorien unterteilt sind:

1. Senden von Daten
2. Empfangen von Daten
3. Informationsservice
4. Transformationsschnittstelle zum Generieren von PDF-Dokumenten

3.1.1 Senden von Daten

Das Senden von Daten benötigt zwei Schnittstellenmethoden:

- Anhänge hochladen
  Damit die Schnittstellen nicht zu sehr überladen werden und doch noch generisch sind, werden die Anhänge ausgelagert und müssen separat dem Gateway übergeben werden. Als Rückgabe erhält man eine Universally unique identifier (UUID), welche als Identifikation des Anhangs dient.
- Nachricht senden
  Nach dem die Nachricht validiert wurde und kein Ablehnungsgrund vorliegt, wird die Nachricht zur späteren Verarbeitung vorgesehen und dem Aufrufer eine UUID als Nachrichten - ID übermittelt.

3.1.2 Empfangen von Daten

Das Empfangen von Daten gliedert sich in 2 Kategorien

1. Nachrichten abfragen
   Hier bei wird unterschieden, ob die Nachrichten per ID (Laufnummer) abgefragt werden sollen, oder innerhalb eines Zeitraumes. Als Ergebnis bekommt man eine Liste von Nachrichten mit den IDs und den dazu gehörigen Anhängen, falls welche vorhanden sind.

   - Nachrichten nach ID abfragen
     Es werden alle Nachrichten ab einer ID zurückgegeben
   - Nachrichten nach Datum abfragen
     Es werden alle Nachrichten innerhalb des angegebenen Zeitraumes angegeben. Wird kein Zeitraum angegeben, so wird das heutige Datum als Obergrenze gewählt.
2. Nachrichten empfangen

   - Daten empfangen
     Das ist das Herunterladen der Nachricht. Als Parameter muss die UUID der Nachricht übermittelt werden.
   - Anhang empfangen
     Äquivalent zum Herunterladen der Nachricht

3.1.3 Informationsservice

Das Informationsservice dient den Systemen als Basis für die Informationsgewinnung bezüglich den Strukturen der auszutauschenden Nachrichten und den möglichen beteiligten Partnern.

3.1.4 Transformationsschnittstelle

Die Transformationsschnittstelle muss die Möglichkeit bieten, ein Datenpaket in ein PDF zu konvertieren. Dies soll den Nutzsystemen als Möglichkeit der Dokumentation dienen, um die zu sendenden Daten auch druckbar, bzw. archivierbar in elektronischer, lesbarer Form zu erhalten. Die Schnittstelle soll so gestaltet werden, dass die Daten auch in andere Formate transformiert werden können.

3.2 Europäische Schnittstellen

Da derzeit die europäischen Schnittstellen noch nicht exakt spezifiziert sind, soll das System so vorgesehen werden, dass die Abfragen dieser Schnittstellen in die Stapelverarbeitung vorzusehen sind. Wie diese Anbindung genau zu erfolgen hat, wird noch spezifiziert und ist nicht Teil dieser Aufgabenstellung.

3.3 Stapelverarbeitung

Die Stapelverarbeitung (Batchverarbeitung) soll in der Nacht, alle Datenpaket abarbeiten und versenden. So soll gewährleistet werden, dass der Server nicht überlastet wird. Da alle Nachrichten nicht zeitkritisch sind, können diese in einem separaten Lauf in der Nacht verarbeitet und versendet werden. Sollten einzelne Nachrichten trotzdem sofort verarbeitet werden müssen, so ist dies in der Konfiguration vorzusehen und asynchron am Serverteil abzuarbeiten.
Da es derzeit noch alte elektronische Formate gibt, so sind diese über die Batchschnittstelle einzulesen, zu verarbeiten und den Endkunden im österreichischen Format als Download zur Verfügung zu stellen.

3.4 Allgemeine Anforderungen

Das System soll so ausgelegt werden, dass es fähig ist, mit Mandanten umzugehen. Das heißt, dass die Nachrichten nicht einzelnen Benutzern zugeteilt werden sondern Institutionen und die Benutzer als Stellvertreter für die Institutionen agieren. Wer diese Nachrichten abholen darf, wird in einer späteren Iteration festgelegt. Dazu muss das System so flexibel ausgelegt werden, dass das nachträgliche Einfügen einer Sicherheitsschicht möglich ist ohne das System komplett neu zu schreiben.

3.5 Abgrenzung

Da die Schnittstelle zur EU noch nicht bekannt ist, werden alle Konvertierungen zum EESSI-System nicht innerhalb dieser Arbeit beschrieben. Es werden nur 2 Konvertierungsvorgänge implementiert und zwar folgende Konvertierungen:

- Konvertierung in PDF
- Konvertierung in E125[14]

Des weiteren wird die Problematik der Anhänge zwar in der Phase der Analyse berücksichtigt, aber nicht implementiert, da dies einen zu großen Einblick in die Unternehmensstruktur geben würde und die Komplexität des Systems dadurch erhöht wird.

[14] Das E125 - Format ist ein auf einem Papierformular basierendes Fixlängenformat

4 Anforderungsanalyse

In der Anforderungsanalyse werden die einzelnen Anforderungen aus der Aufgabenstellung weiter unterteilt und näher untersucht.
Die Anforderungsanalyse wird mittels EA umgesetzt. Dazu muss die Software installiert werden und ist ab diesem Zeitpunkt voll funktionsfähig. Als weiteren Schritt muss ein neues Projekt angelegt werden, welches alle Elemente und Diagramme enthält. Da die Einrichtung des Generierens der Dokumentation mittels EA zum jetzigen Zeitpunkt zu viel Zeit in Anspruch nehmen würde, wird darauf verzichtet und es werden lediglich die Diagramme mittels EA modelliert und als Bild exportiert.

4.1 Systemlandschaft

Das EESSI-System wird als verteiltes System implementiert, sodass die Staaten unabhängig untereinander Senden und Empfangen können. Der Ansatz mit einem zentralen Server, welche die Nachrichten verteilt, wurde bereits getestet und als unzureichend befunden, da der „Bottle-Neck“ - der zentrale Server - gleich bei den ersten Tests in die Knie gezwungen wurde, was einer DOS-Attacke[15] gleichkam.

[15] Denial of Service - Attacke

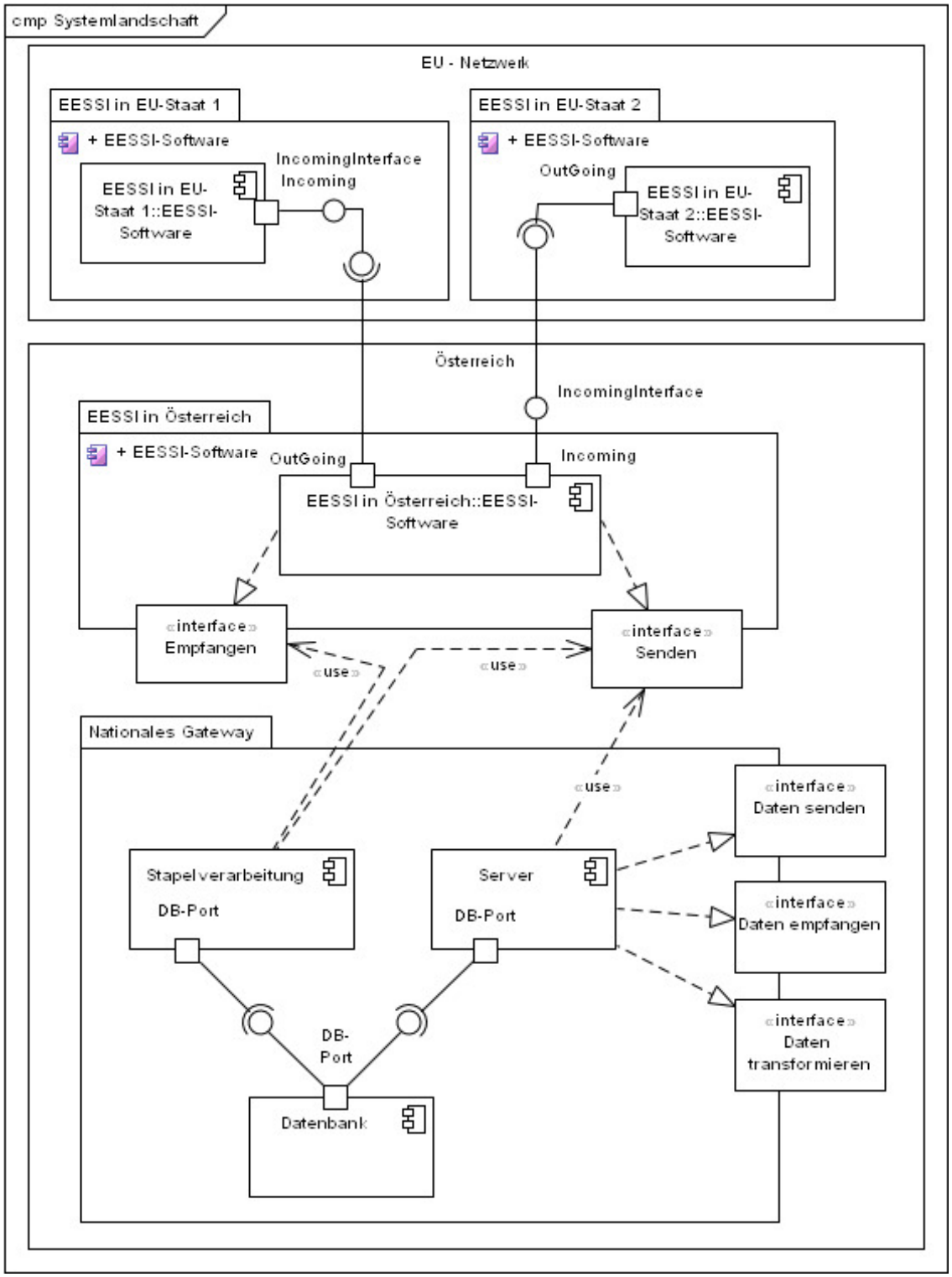

Abbildung 4.1: Systemlandschaft

Das Nationales Gateway (NG) dient hierbei zur Ankopplung an das EESSI-System und holt die Nachrichten mittels Stapelverarbeitung vom EESSI-System ab und sendet die Nachrichten mit Massendaten, bzw. Nachrichten, welche während der Verarbeitung einen Fehler erzeugt haben. Der Serverteil des NGs kann die kleinen Nachrichten auch

direkt an das EESSI-System versenden.

4.2 Anforderungen

Index	Beschreibung
ANF001	Jede Anfrage an das System muss vom System validiert werden
ANF002	Ein Anhang darf nur ein PDF sein
ANF003	Die anfragende Stelle muss ein VT sein
ANF004	Die Nachricht muss validiert werden
ANF005	Der Anhang muss validiert werden
ANF006	Im Falle eines Validierungsfehlers muss eine entsprechende Fehlermeldung zurückgegeben werden
ANF007	Alle gespeicherten Elemente (Nachrichten, Anhänge, transformierte Nachrichten) werden mittels UUID referenziert
ANF008	Ein VT darf nur für ihn bestimmte Nachtrichten sehen
ANF009	Ein VT darf nur für ihn bestimmte Nachtrichten herunterladen
ANF010	Eine übergebene UUID muss validiert werden, dass es eine UUID ist
ANF011	Der Absender muss im Verzeichnisdienst eingetragen sein
ANF012	Eine Nachricht muss an mindesten einen Empfänger gerichtet sein
ANF013	Eine Nachricht kann an mehrere Empfänger gesendet werden
ANF014	Ein Empfänger muss im Verzeichnisdienst eingetragen sein
ANF015	Die Stapelverarbeitung muss stabil laufen
ANF016	Jeder einzelne Schritt der Stapelverarbeitung soll separat getriggert werden können
ANF017	Jeder einzelne Schritt der Stapelverarbeitung muss in sich gekapselt sein
ANF018	Ein Wiederaufsetzen der Verarbeitung muss möglich sein

Tabelle 4.1: Anforderungstabelle

4.2.1 ANF001

Die SOAP-Nachricht, welche an das System übergeben wird muss validiert werden, damit keine syntaktischen Fehler im Aufruf enthalten sein können.

4.2.2 ANF002

Da die Nachrichten auch als PDF zur Verfügung gestellt werden, darf der Anhang nur ein PDF sein.

4.2.3 ANF003

Alle Anfragen müssen von einem gültigen österreichischer Sozialversicherungsträger (VT) kommen.

4.2.4 ANF004

Die Nachricht, die Nutzdaten, wird als binäres Objekt an die SOAP-Anfrage angehängt und separat validiert werden.

4.2.5 ANF005

Die Liste der Anhänge muss validiert werden, ob die Anhänge auch schon am NG verfügbar sind und noch nicht gelöscht wurden.

4.2.6 ANF006

Wenn eine Nachricht nicht valide ist, śo muss die Antwort den Validierungsfehler enthalten.

4.2.7 ANF007

Die primären Datenschlüssen werden mittels UUID abgebildet.

4.2.8 ANF008

Jeder darf nur die Nachrichten sehen, welche für ihn bestimmt sind.

4.2.9 ANF009

Jeder darf nur die Nachrichten herunterladen, welche für ihn bestimmt sind.

4.2.10 ANF010

Nicht jede Kombination im Format einer UUID ist auch eine UUID, weshalb die UUID validiert werden muss.

4.2.11 ANF011

Alle möglichen Institutionen stehen im Verzeichnisdienst der EU, weshalb auch nur Institutionen, welche im Verzeichnisdienst eingetragen sind als Sender verwendet werden dürfen.

4.2.12 ANF012

Eine Nachricht muss mindestens einen Empfänger, welcher auch im Verzeichnisdienst eingetragen ist.

4.2.13 ANF013

Da manche Nachrichten auch an mehrere Empfänger gerichtet sein können, muss es möglich sein, auch eine Liste an Empfängern zu übergeben.

4.2.14 ANF014

Alle Empfänger müssen im Verzeichnisdienst eingetragen sein.

4.2.15 ANF016

Jeder einzelne Schritt der Stapelverarbeitung soll separat getriggert werden können, d.h., dass es einen Einstiegspunkt geben soll, damit nur dieser Schritt mittels Bat- oder Shell-Script gestartet werden kann, falls Bedarf dafür vorliegt.

4.2.16 ANF017

Jeder einzelne Schritt der Stapelverarbeitung muss in sich gekapselt sein, damit, falls die gesamte Kette auf einmal gestartet wird, auch die Ablaufkette durchlaufen werden kann, ohne dass diese dazwischen unterbrochen wird.

4.2.17 ANF018

Sollte die Verarbeitung einer Nachricht fehlschlagen, so muss diese beim nächsten Lauf ausgelassen werden können, solange, bis der Fehler behoben wurde und die Verarbeitung gelingen kann.

4.3 Anwendungsfälle für Benutzer

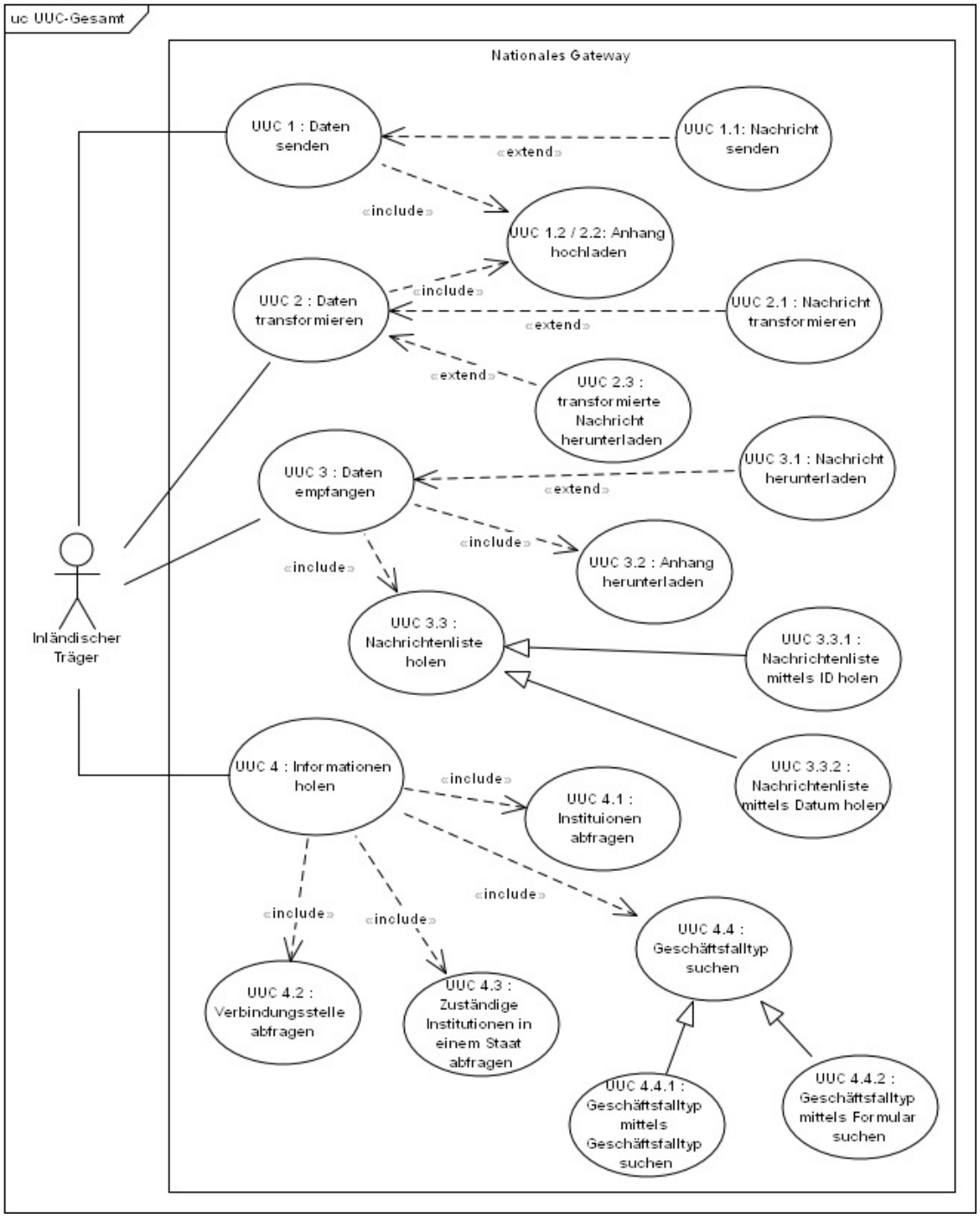

Abbildung 4.2: UC: Gesamtübersicht Benutzer

Diese UCs betreffen alle österreichischen Benutzer des NGs.

Index	Name	Kurzbeschreibung
UUC 1	Daten senden	Der Anwender möchte Daten an die EU senden
UUC 2	Daten transformieren	Der Anwender möchte die Daten als PDF für die Dokumentation transformieren
UUC 3	Daten empfangen	Der Anwender möchte seine, vom Ausland empfangenen Nachtrichten abholen
UUC 4	Informationen holen	Der Anwender möchte sich Informationen über das Ablaufmodell holen

Tabelle 4.2: Primäre Anwendungsfälle für Anwender / Partnersysteme

4.3.1 UUC 1 : Daten senden

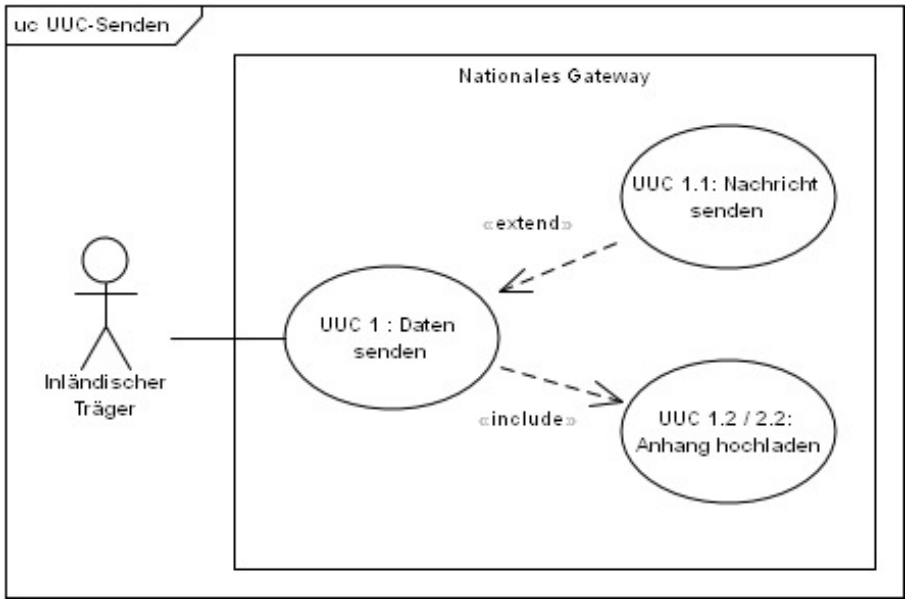

Abbildung 4.3: UC: Daten senden

Der Anwender möchte eine Nachricht an eine Institution innerhalb der EU senden. Dabei kann er nur die Daten versenden, oder er möchte auch noch einen, oder mehrere Anhänge dazu geben, welche im Format PDF übergeben werden müssen. Da das System ohne Freigabemechanismus arbeitet, müssen die Anhänge zuerst übergeben werden und im Anschluss in der Nachricht referenziert werden.

4.3.1.1 UUC 1.1 : Nachricht senden

Der Anwender über gibt dem System über die SOAP-Schnittstelle die Nachricht im Format, welches im Anhang A.1.1 beschrieben ist. Die übergebenen Daten werden validiert und im Erfolgsfall erhält der Aufrufer eine UUID[16], welche die Nachricht eindeutig

[16] Siehe Anhang A.1.2

identifiziert. Im Fehlerfall wir eine Fehlermeldung zurückgegeben, welche den Fehler beschreibt[17].

4.3.1.2 UUC 1.2 : Anhang hochladen

Zu einer Nachricht sollen Anhänge, PDF-Dokumente, mit versendet werden. Dazu muss der Anwender den Anhang hochladen[18] und erhält eine UUID[19]. Mit dieser UUID muss der Anhang innerhalb der Nachricht referenziert werden.

4.3.2 UUC 2 : Daten transformieren

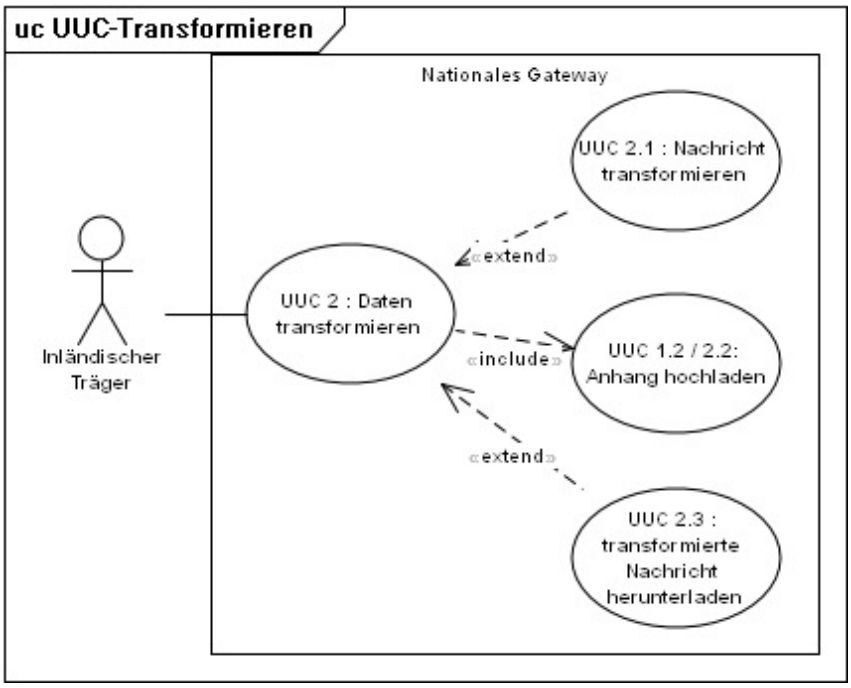

Abbildung 4.4: UC: Daten transformieren

Der Anwender hat die Möglichkeit, dass er die von ihm zu sendenden Daten auch als PDF transformieren lassen kann und dieses PDF im Anschluss sich ausdrucken und zu den Akten legen kann.

4.3.2.1 UUC 2.1 : Nachricht transformieren

Der Anwender über gibt dem System über die SOAP-Schnittstelle die Nachricht im Format, welches in Anhang A.1.26 beschreiben ist. Die übergebenen Daten werden vali-

[17] Siehe Anhang A.1.10
[18] Siehe Anhang A.1.8
[19] Siehe Anhang A.1.9

diert und im Erfolgsfall erhält der Aufrufer eine UUID[20], welche die Nachricht eindeutig identifiziert. Im Fehlerfall wir eine Fehlermeldung zurückgegeben, welche den Fehler beschreibt[21].

4.3.2.2 UUC 2.2 : Anhang hochladen

Siehe dazu 4.3.1.2, Punkt UC 1.2

4.3.2.3 UUC 2.3 : transformierte Nachricht herunterladen

Wenn die Nachricht vom System transformiert wurde, kann man mittels der zuerst erhaltenen UUID und den Parametern für die Transformation die transformierte Nachricht herunterladen[22] und erhält das fertige PDF[23]. Sollte die Nachricht noch nicht transformiert worden sein, so erhält man eine Fehlermitteilung, dass die Nachricht noch nicht fertig ist[24].

4.3.3 UUC 3 : Daten empfangen

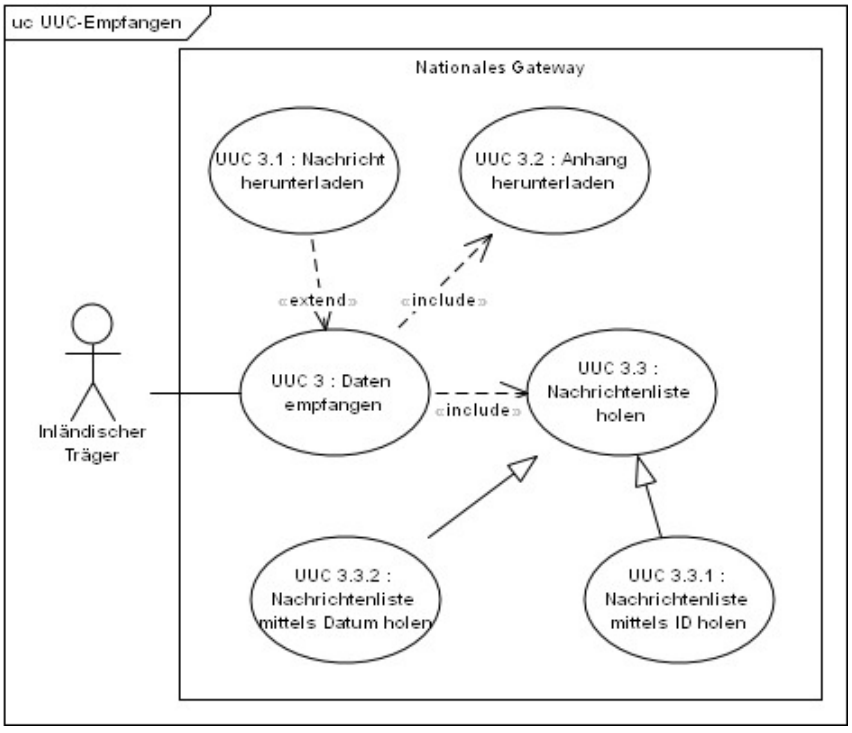

Abbildung 4.5: UC: Daten Empfangen

[20] Siehe Anhang A.1.27
[21] Siehe Anhang A.1.10
[22] Siehe Anhang A.1.28
[23] Siehe Anhang A.1.29
[24] Siehe Anhang A.1.10

Der UC „Daten empfangen“ beinhaltet alle Aktionen, welche dem Anwender dazu dienen, Nachrichten vom NG abzuholen.

4.3.3.1 UUC 3.1 : Nachricht herunterladen

Der Anwender kann mittels angegebener UUID die Nachricht herunterladen[25] und erhält die Nachricht[26].

4.3.3.2 UUC 3.2 : Anhang herunterladen

Der Anwender kann mittels angegebener UUID den Anhang herunterladen[27] und erhält den Anhang[28].

4.3.3.3 UUC 3.3 : Nachrichtenliste holen

Dies ist ein abstrakter UC, da man entweder die Nachrichtenliste mittels ID, oder Datum holen kann.
Das Resultat ist immer eine Liste an Nachrichten, welche dem Datentyp aus Anhang [29] entspricht.

4.3.3.4 UUC 3.3.1 : Nachrichtenliste mittels ID holen

Dazu sendet die anfragende Stelle die Daten im Format, welches im Anhang [30] beschrieben ist.

4.3.3.5 UUC 3.3.2 : Nachrichtenliste mittels Datum holen

Dazu sendet die anfragende Stelle die Daten im Format, welches im Anhang [31] beschrieben ist.

[25] Siehe Anhang A.1.11
[26] Siehe Anhang A.1.12
[27] Siehe Anhang A.1.6
[28] Siehe Anhang A.1.7
[29] Siehe Anhang A.1.15
[30] Siehe Anhang A.1.13
[31] Siehe Anhang A.1.14

4.3.4 UUC 4 : Informationen holen

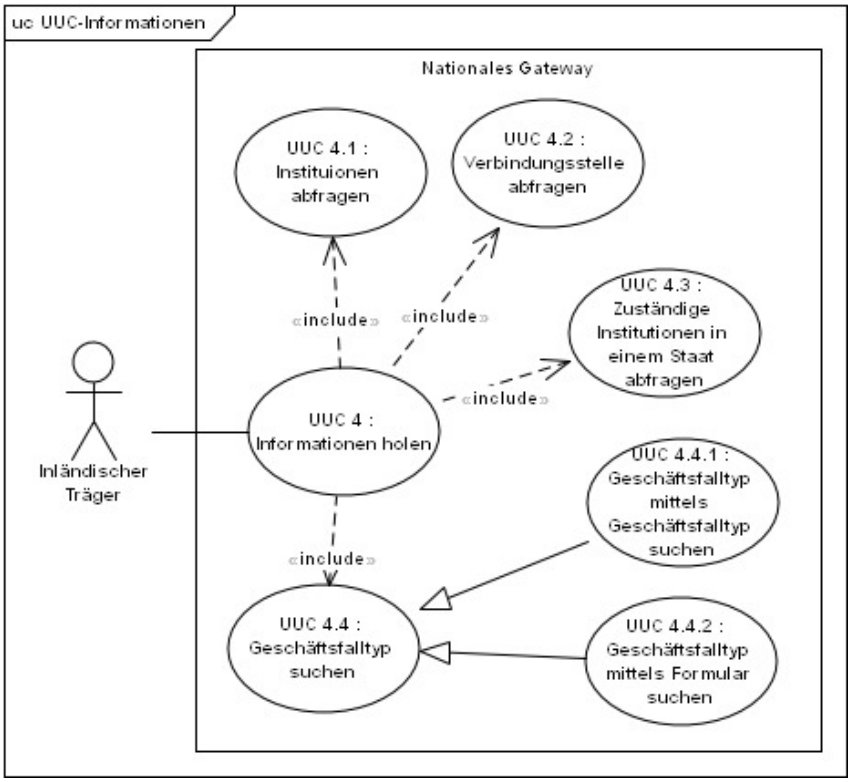

Abbildung 4.6: UC: Informationen holen

Der UC „Informationen holen“ dient dem Anwender dazu, Informationen über die möglichen ausländischen Partner, die Verbindungsstellen der einzelnen Staaten, die Geschäftsfalltypen[32] und die Formulare zu erhalten.
Generell werden die Staaten innerhalb des EESSI-Systems mittels des 2-stelligen Staatencodes nach ISO-3166-1 adressiert[33]

4.3.5 UUC 4.1 : Institutionen abfragen

Um Informationen zu den einzelnen Institutionen zu erhalten, muss man die Daten nach dem Format, welches in A.1.17 beschrieben ist, senden. Dann erhält man eine Liste von Institutionen nach dem Format aus A.1.19.

[32] Der Geschäftsfalltyp ist ein Container, welche alle notwendigen und möglichen Formulare enthält und einen Geschäftsfall einzuleiten und diesen auch abzuschließen. Ein Formular kann nur in einem Geschäftsfalltyp vorkommen.
[33] Vergleiche dazu A.1.5

4.3.6 UUC 4.2 : Verbindungsstelle abfragen

Für gewisse Geschäftsfalltypen ist nur eine Stelle als SPoC in einem Staat zuständig. Man erhält diese Institution, in dem man die Anfrage nach dem Format aus A.1.20 sendet und erhält die Antwort im Format aus A.1.21. Hierbei kann man entweder eine Institution, oder auch keine Institution erhalten, wenn entweder keine Institution eingetragen wurde, bzw. dieser Geschäftsfalltyp für mehrere Institutionen zulässig ist.

4.3.7 UUC 4.3 : Zuständige Institutionen in einem Staat abfragen

Hierbei kann der Anwender zuständige Institutionen suchen, welche einen Geschäftsfalltyp empfangen können. Auf die Anfrage[34] erhält man eine Liste nach dem Format aus A.1.19.

4.3.8 UUC 4.4 : Geschäftsfalltyp suchen

Die Abfragen nach Informationen zu einem Geschäftsfalltyps kann auf 2 Arten erfolgen. Entweder kennt man die GeschäftsfalltypID und die GeschäftsfalltypVersion oder man kennt eine FormularID und die FormularVersion.
Als Antwort erhält man den Geschäftsfalltyp mit der Liste an Formularen[35]. Die Formulare enthalten auch die Informationen, welches Formular von welcher Partei gesendet werden darf und welches Formular das Startformular[36] ist.

4.3.8.1 UUC 4.4.1 : Geschäftsfalltyp mittels Geschäftsfalltyp suchen

Aufbau der Anfrage nach A.1.22

4.3.8.2 UUC 4.4.2 : Geschäftsfalltyp mittels Formular suchen

Aufbau der Anfrage nach A.1.23

[34] Vergleiche dazu A.1.18
[35] Vergleiche dazu A.1.24
[36] Oder auch Initialformular

4.4 Anwendungsfälle für Stapelverarbeitung

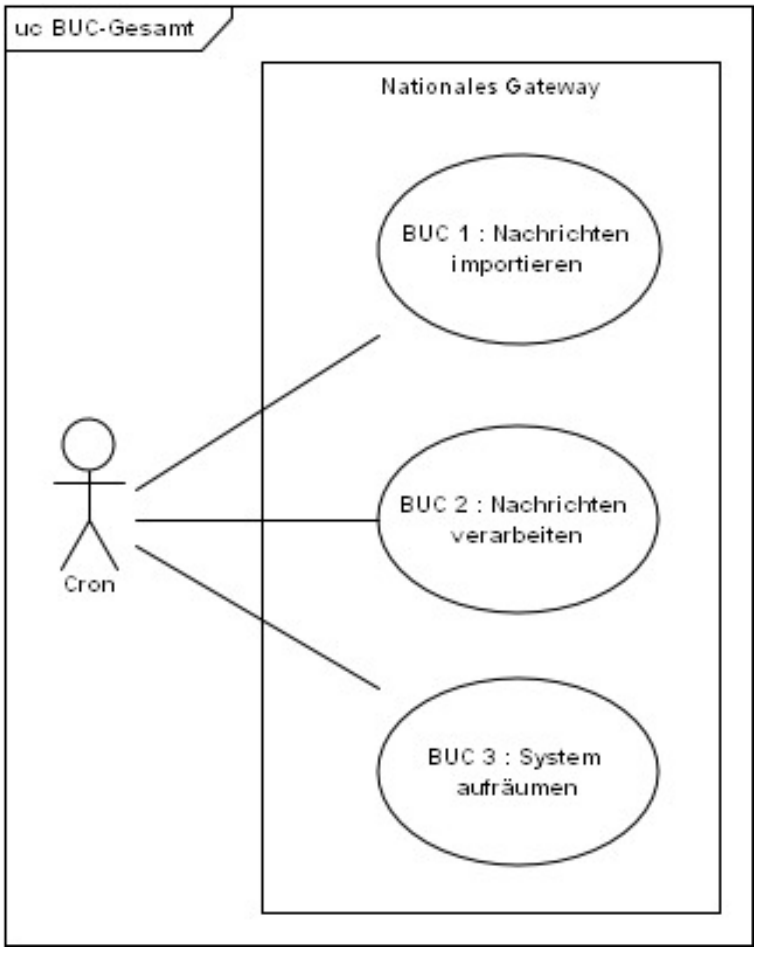

Abbildung 4.7: UC: Gesamtübersicht Stapelverarbeitung

Diese UCs betreffen die Stapelverarbeitung.

Index	Name	Kurzbeschreibung
BUC 1	Nachrichten importieren	Die Stapelverarbeitung soll die Datenquellen nach möglichen Nachrichten durchsuchen
BUC 2	Nachrichten verarbeiten	Die Stapelverarbeitung soll die importierten Nachrichten verarbeiten
BUC 3	System aufräumen	Die Stapelverarbeitung soll alte nicht mehr benötigte Daten löschen

Tabelle 4.3: Primäre Anwendungsfälle für die Stapelverarbeitung

4.4.1 BUC 1 : Nachrichten importieren

Die Stapelverarbeitung soll alle möglichen Datenquellen durchsuchen und die gefunden Nachrichten in das System aufnehmen, damit diese zu einem späteren Zeitpunkt verarbeitet werden können.

4.4.2 BUC 2 : Nachrichten verarbeiten

Die Stapelverarbeitung soll alle noch nicht verarbeiteten, fehlerfreien Nachrichten bearbeiten. Dabei gibt es 2 Szenarien. Eine Nachricht aus dem Ausland soll verarbeitet werden, oder eine Nachricht aus dem Inland soll versendet werden.
Ausländische Nachricht bearbeiten

1. Nachricht duplizieren: Die Nachricht wird für jeden Empfänger in einen eigenen Ordner auf der Festplatte kopiert
2. Nachricht transformieren: Die Nachricht wird in das Format für den Empfänger transformiert.
3. Nachricht zum Download bereitstellen: Die Nachricht wird dem Empfänger zum Download angeboten.
4. Nachricht abschließen: Die Nachricht wird als erledigt markiert und in das Protokoll eingetragen.

Inländische Nachricht bearbeiten

1. Nachricht duplizieren: Die Nachricht wird für jeden Empfänger in einen eigenen Ordner auf der Festplatte kopiert
2. Nachricht transformieren: Die Nachricht wird in das Format für den Empfänger transformiert.
3. Nachricht transportieren: Die Nachricht wird dem Empfänger zugestellt.
4. Nachricht abschließen: Die Nachricht wird als erledigt markiert und in das Protokoll eingetragen.

4.4.3 BUC 3 : System aufräumen

Die Stapelverarbeitung soll alle nicht mehr benötigten Dateien löschen und so für Speicherplatz sorgen. Dies bedeutet, dass die temporären Daten, welche während der Transformation von Nöten waren nach Ablauf einer gewissen Zeitspanne gelöscht werden.

4.5 Ablaufbeschreibungen

Innerhalb dieses Abschnitts werden die einzelnen Abläufe beschrieben, damit ein gemeinsames Verständnis der Vorgänge erhält.

4.5.1 Ablaufbeschreibungen der Anwendungsfälle

4.5.1.1 UUC 1 : Daten senden

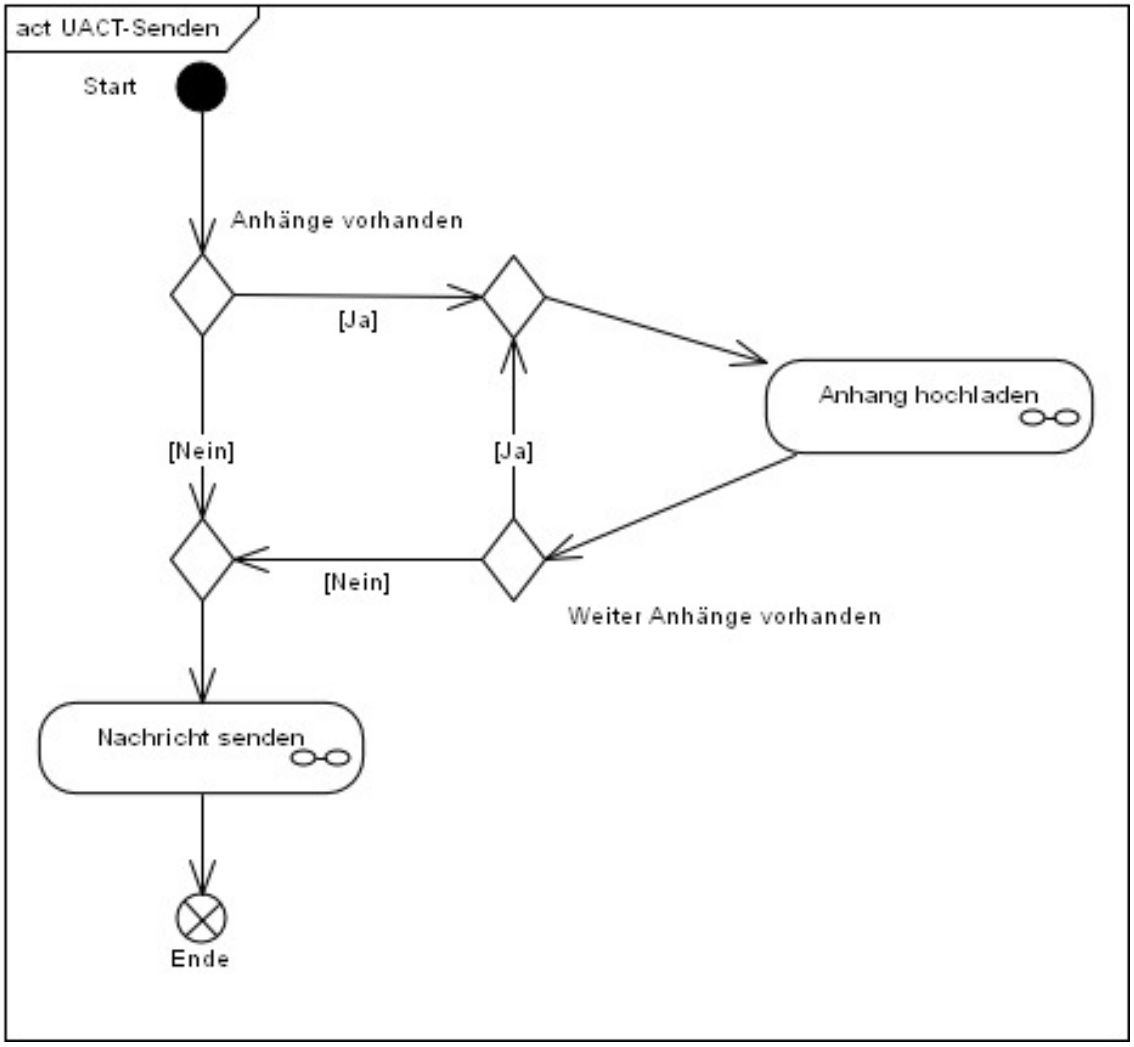

Abbildung 4.8: Benutzerablauf: Daten senden

Damit ein Anwender seine Daten an einen ausländischen Partner senden kann, müssen zuerst die Anhänge dem System übergeben werden[37], welche in der Nachricht referenziert werden müssen und im Anschluss muss noch die Nachricht selbst übergeben

[37] Vergleiche Systemablauf Anhang hochladen 4.5.3.1

werden[38].

4.5.1.2 UUC 2 : Daten transformieren

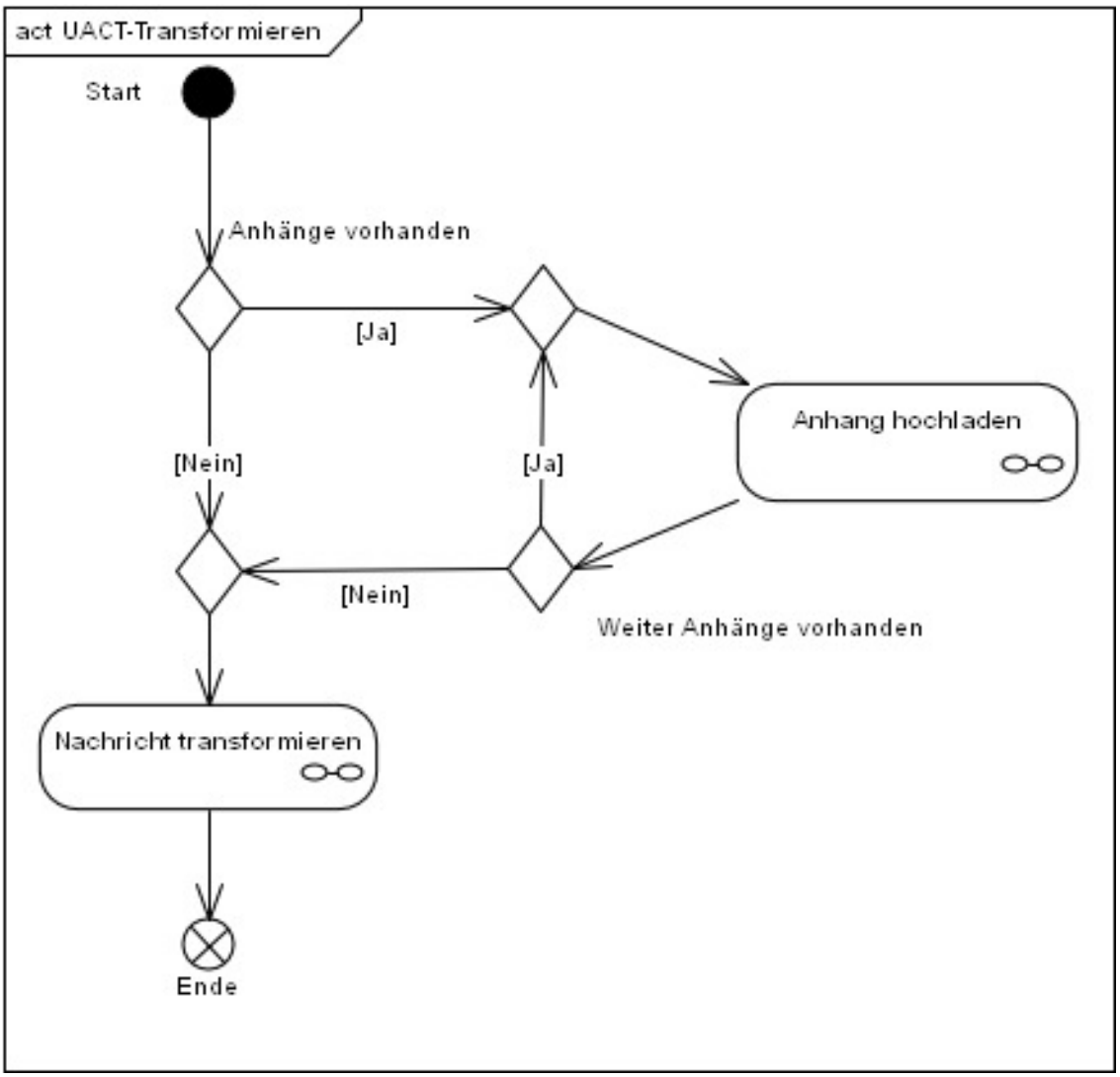

Abbildung 4.9: Benutzerablauf: Daten transformieren

Damit ein Anwender seine Daten transformieren kann, müssen zuerst die Anhänge dem System übergeben werden[39], welche in der Nachricht referenziert werden müssen und im Anschluss muss noch die Nachricht selbst übergeben werden[40].

[38] Vergleiche Systemablauf Nachricht senden 4.5.3.2
[39] Vergleiche Systemablauf Anhang hochladen 4.5.3.1
[40] Vergleiche Systemablauf Nachricht transformieren 4.5.3.3

4.5.1.3 UUC 3 : Daten empfangen

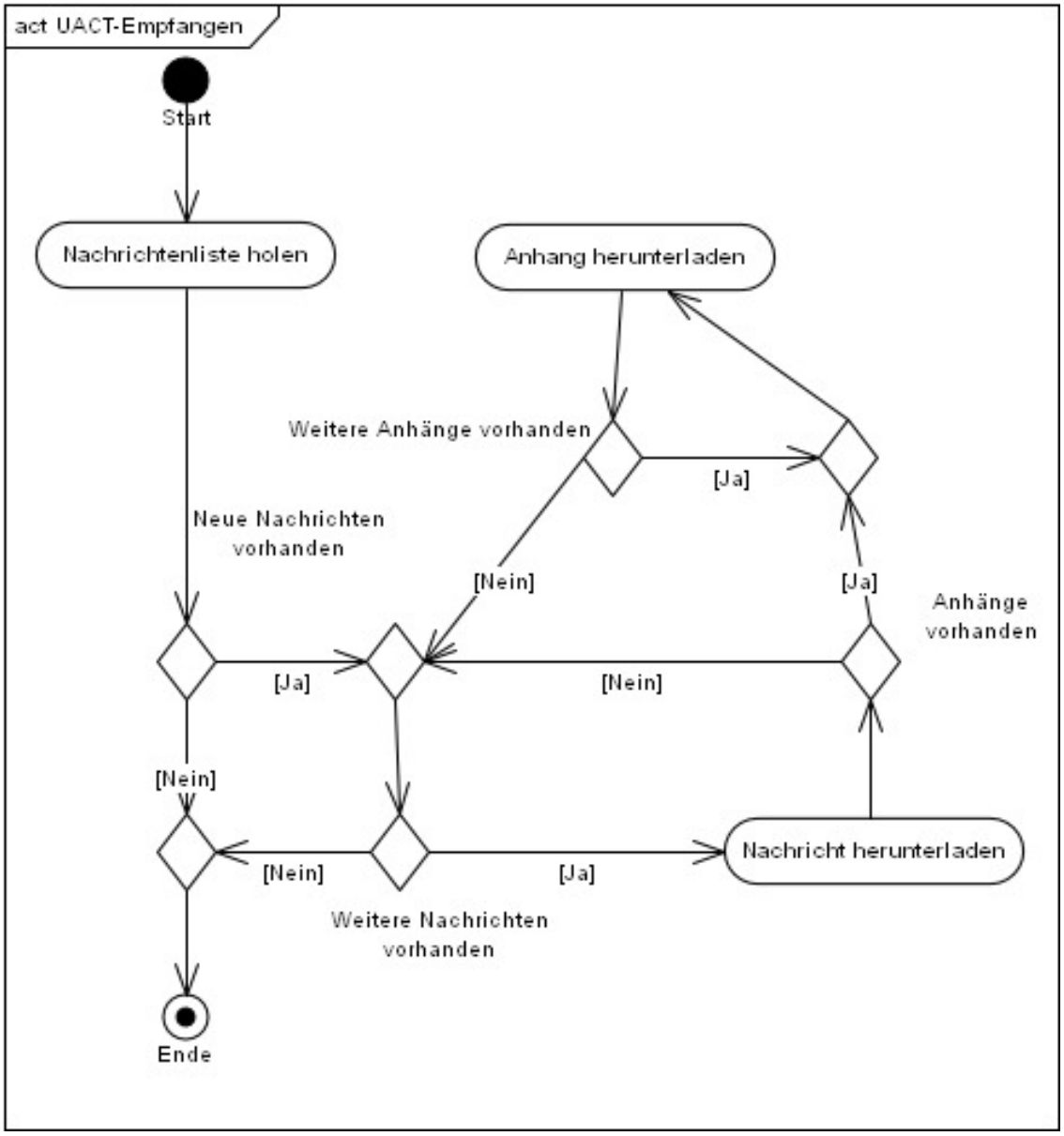

Abbildung 4.10: Benutzerablauf: Daten empfangen

Wenn Daten empfangen werden sollen, so muss in einem ersten Schritte die Metadaten der Nachrichten abgefragt werden.
Die Metadaten einer Nachricht enthalten die Werte, welche in Abschnitt 4.3.3.3 beschrieben sind. Mittels dieser Werte können nun die Anhänge und die Nachricht selbst geholt werden.

4.5.1.4 UUC 4 : Informationen holen

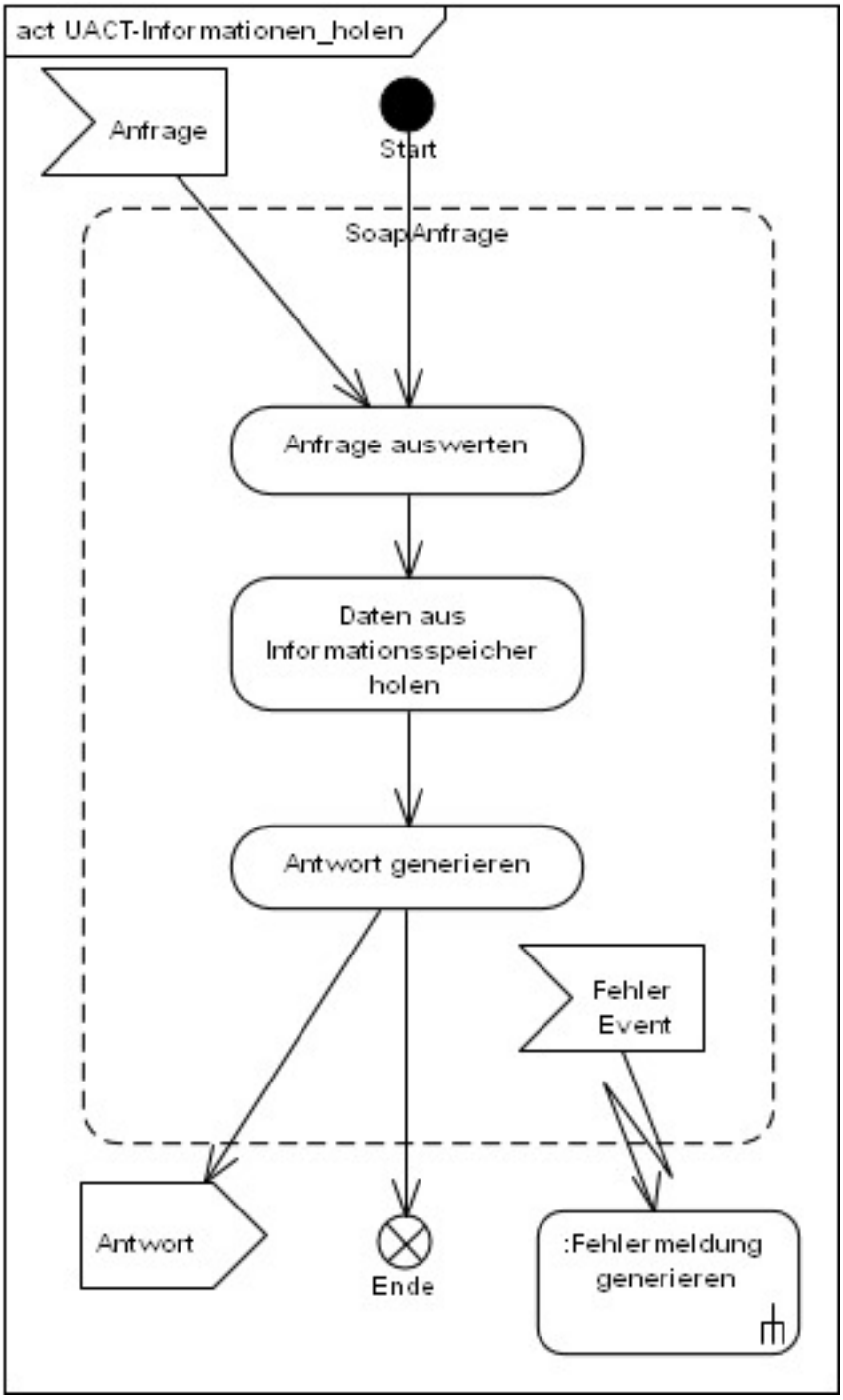

Abbildung 4.11: Benutzerablauf: Informationen holen

Die UCs der Gruppe UUC 4 sind reine Anfragen an das dahinter liegende Informationssystem und sind für alle frei verfügbar. Der Ablauf ist für alle gleich weshalb diese nicht separat angeführt werden.

4.5.2 Abläufe der Stapelverarbeitung

4.5.2.1 BUC 1 : Nachrichten importieren

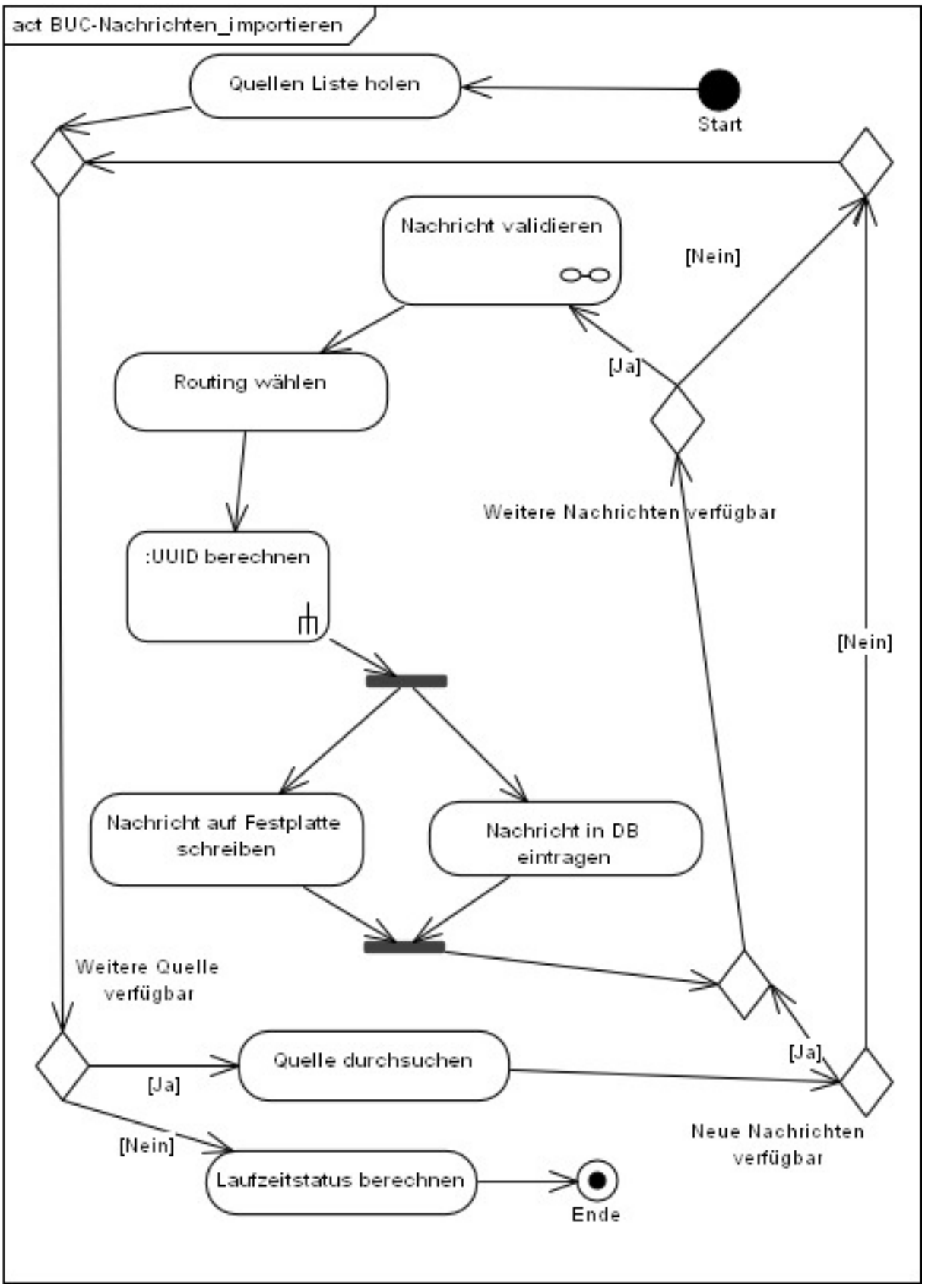

Abbildung 4.12: Stapelverarbeitung: Nachrichten importieren

Zuerst wird die Liste der möglichen Datenquellen eingelesen und jede Quelle befragt, ob es neue Nachrichten gibt. Wenn neue Nachrichten verfügbar sind so werden diese einzeln dem System übergeben und in die Datenbank und auf die Festplatte geschrieben, damit diese in einem späteren Schritt verarbeitet werden können.

4.5.2.2 BUC 2 : Nachrichten verarbeiten

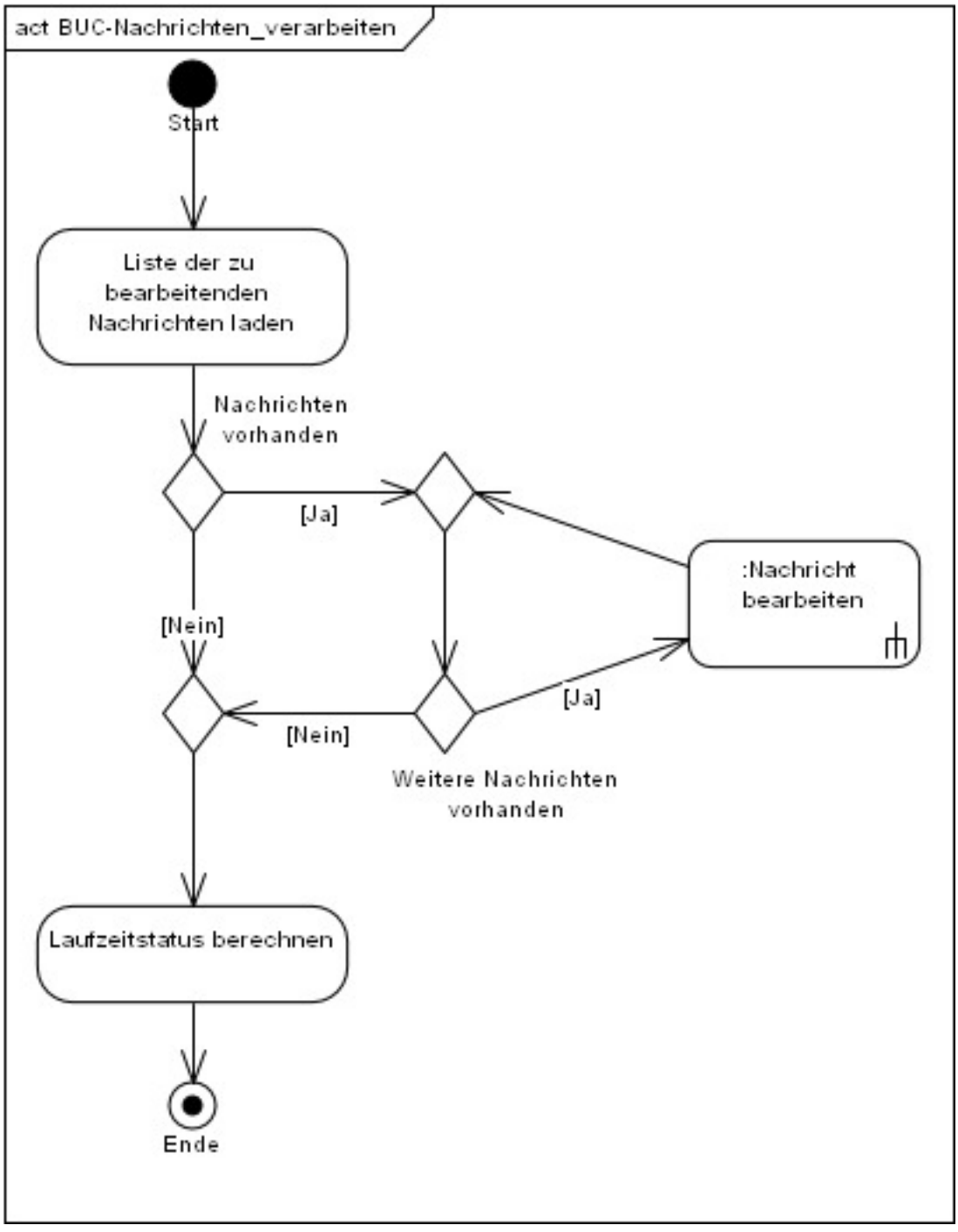

Abbildung 4.13: Stapelverarbeitung: Nachrichten verarbeiten

Es werden die noch nicht fertig abgearbeiteten Nachrichten aus der Datenbank gelesen und im Anschluss verarbeitet.

4.5.2.3 BUC 3 : System aufräumen

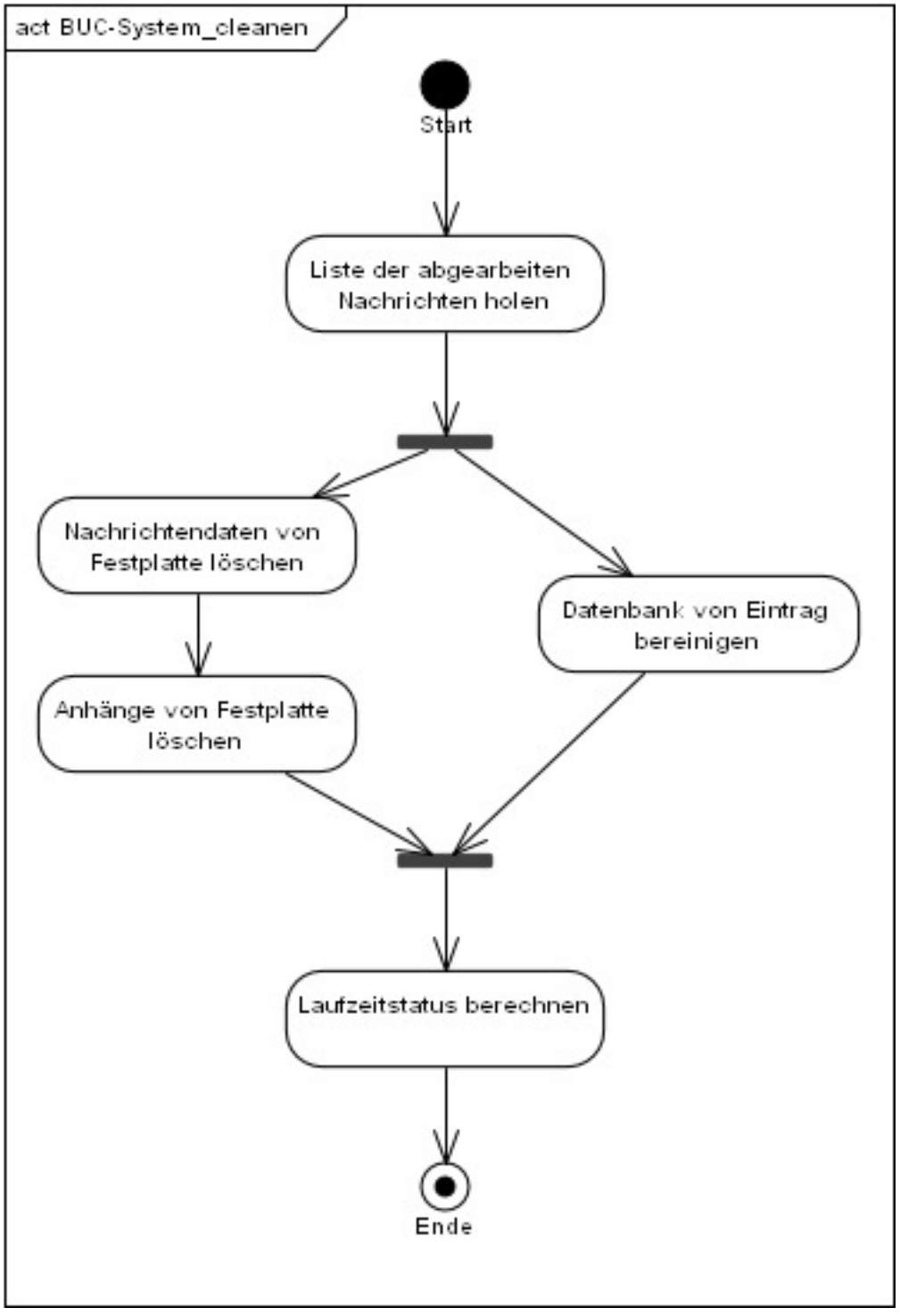

Abbildung 4.14: Stapelverarbeitung: System aufräumen

Die Daten werden zur besseren Verarbeitung temporär auf die Festplatte gespeichert. Wenn die Daten verarbeitet worden sind und keine weiteren Anfragen dazu gekommen sind, so können diese nach Ablauf einer Frist, welche noch bestimmt werden muss, gelöscht werden. Dies betrifft nur die Eingabedaten. Die auszugebenden Daten sind derzeit von diesem Prozess nicht betroffen.

4.5.3 Systemabläufe

4.5.3.1 Anhang hochladen

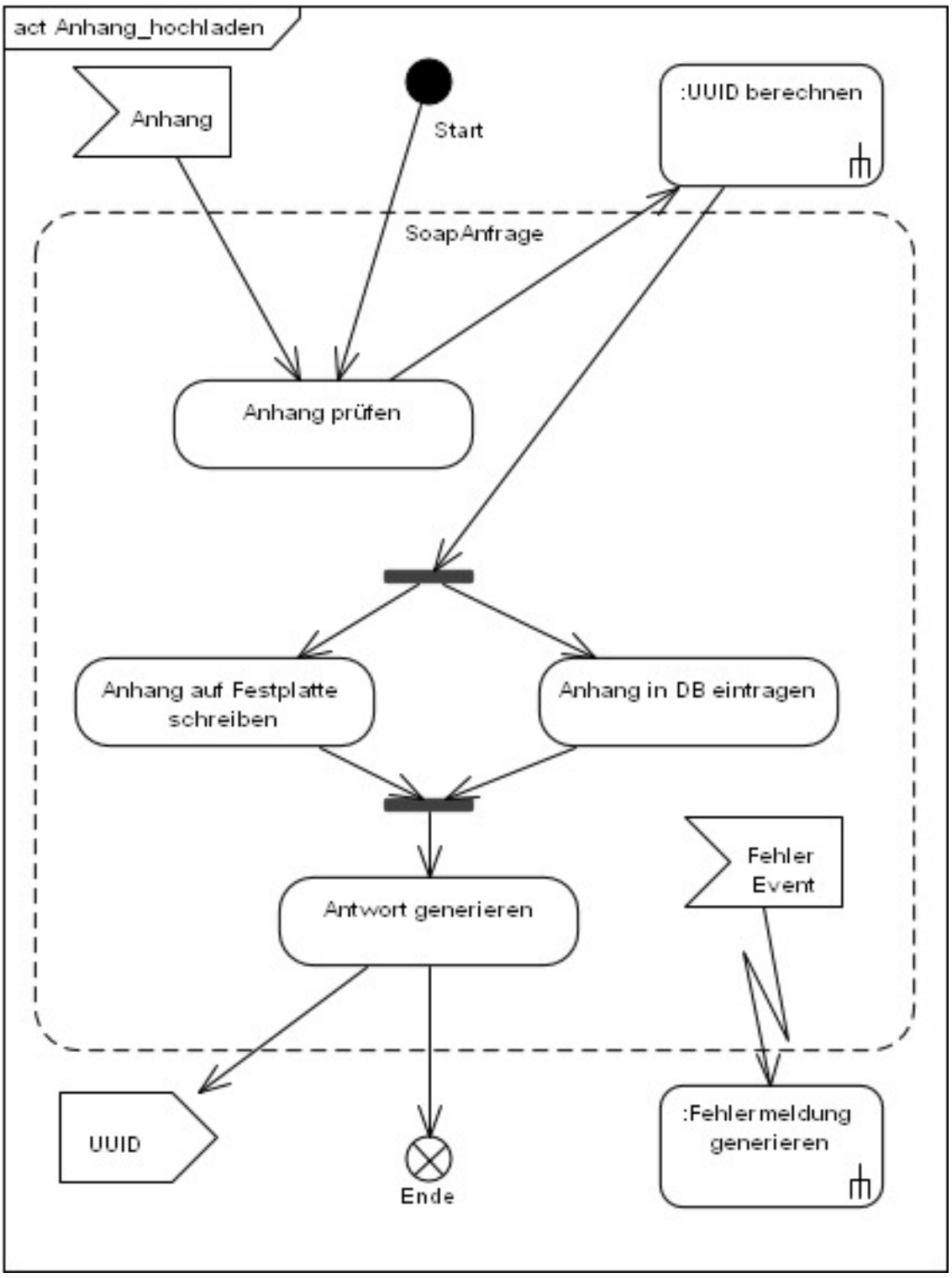

Abbildung 4.15: Systemablauf: Anhang hochladen

Das Diagramm 4.15 beschreibt den Ablauf, den das System umsetzen muss, wenn ein Anhang hochgeladen wird. Dazu muss der Anhang überprüft werden, ob dieser ein PDF ist. Dann wird eine UUID generiert. Mittels dieser UUID muss der Anhang referenziert werden, sowohl innerhalb der Nachricht als auch in der Liste der Anhänge der SOAP-Nachricht. Mit der generierten UUID wird der Anhang auf die Festplatte geschrieben und in der Datenbank vermerkt. Dann wird Antwort generiert und so der anfragenden Stelle die UUID mitgeteilt.

4.5.3.2 Nachricht senden

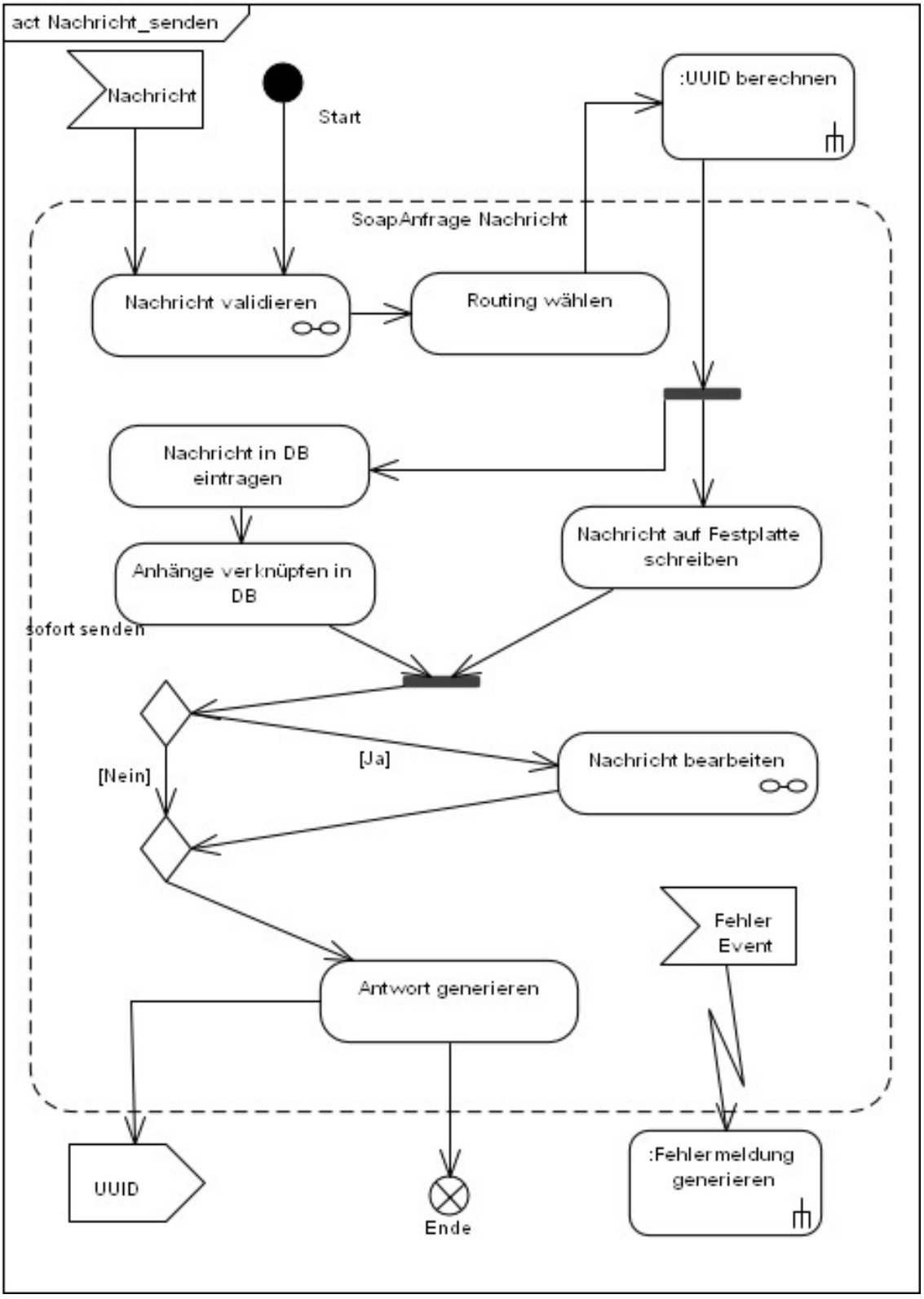

Abbildung 4.16: Systemablauf: Nachricht senden

Wird eine Nachricht dem System zum Senden übergeben, so kommt der Ablauf aus dem Diagramm 4.16 zum tragen. Dabei wird zuerst die Nachricht validiert[41]. Als nächstes muss das Routing gewählt werden, womit die Transformation und der Ausgabekanal bestimmt werden. Nun wird eine UUID generiert, welche der Bestimmung der Nachricht und der Nachvollziehbarkeit dient. Mit der generierten UUID wird die Nachricht nun auf die Festplatte geschrieben und in der Datenbank vermerkt. Des weiteren werden die Anhänge, welche in der Liste der Anhänge der SOAP-Nachricht übergeben wurden mit der Nachricht verknüpft. Wenn die Nachricht keine Massendaten enthält, so kann diese sofort weiter verarbeitet werden[42]. Die Verarbeitung muss asynchron erfolgen, um die Respoinsezeiten so gering wie möglich zu halten. Zum Schluss wird noch die Antwort mit der UUID generiert und dem Aufruf zurückgegeben.

[41] Siehe dazu Systemablauf 4.5.3.8

[42] Die Verarbeitung geschieht nach dem Ablauf, welcher in Kapitel 4.5.3.9 beschrieben ist.

4.5.3.3 Nachricht transformieren

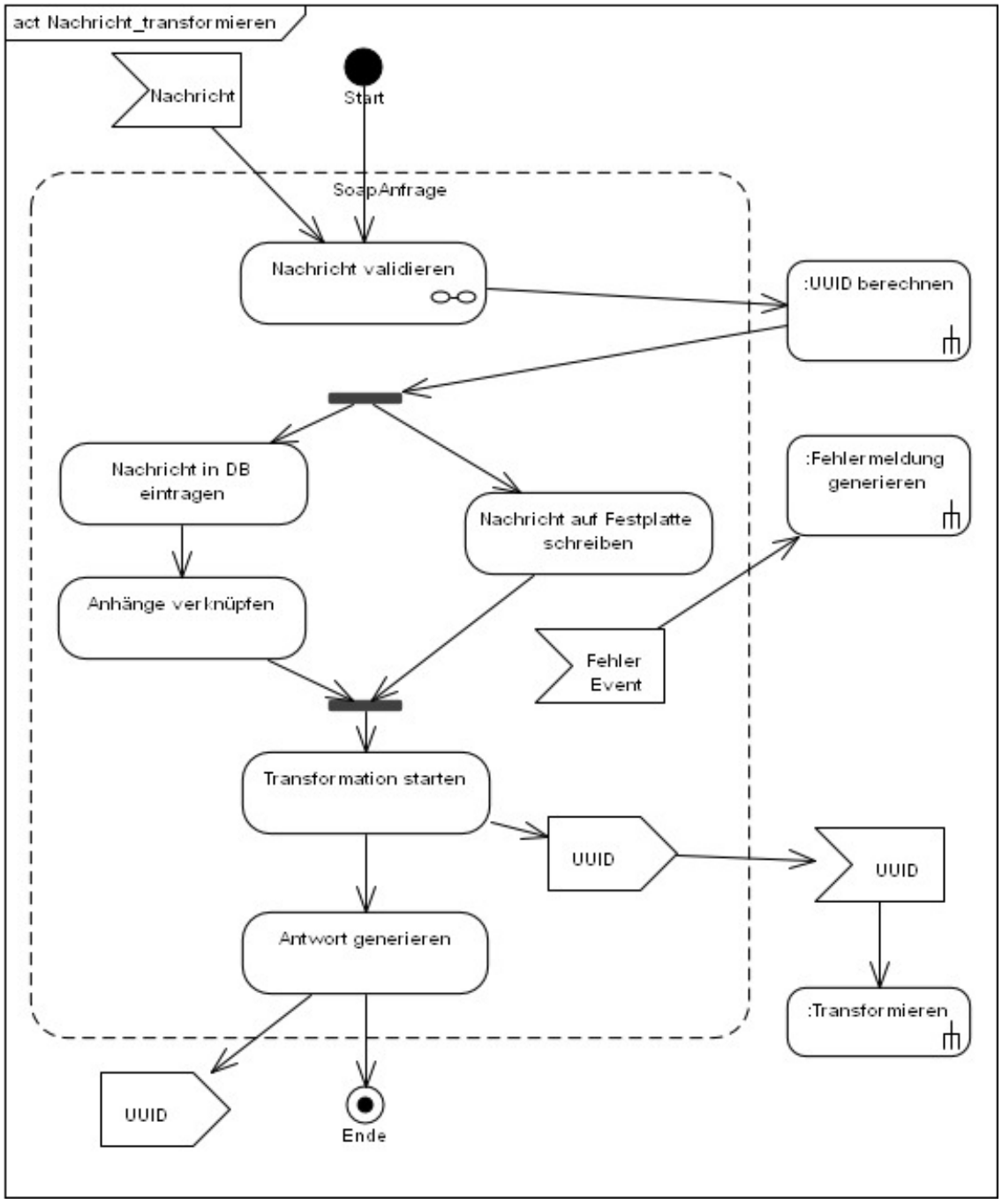

Abbildung 4.17: Systemablauf:Nachricht transformieren

Der Ablauf „Nachricht transformieren“ übernimmt eine Anfrage zum Transformieren in ein PDF. Dazu wird die Nachricht validiert[43] und die UUID für diese Nachricht generiert. Danach wird die Nachricht in der Datenbank eingetragen und allfällige Anhänge werden verknüpft. Zeitgleich wird die Nachricht auf die Festplatte geschrieben, damit diese den

[43] Siehe dazu 4.5.3.8

allgemeinen Lauf nehmen kann. Danach wird die Transformation angestoßen, da diese asynchron abläuft. Die zuerst generierte UUID wir dem Benutzer in der Antwort mitgeteilt. Im Fehlerfall bekommt der Aufrufer eine Fehlermeldung retourniert, welche auf den Fehler hinweist.

4.5.3.4 Duplizieren

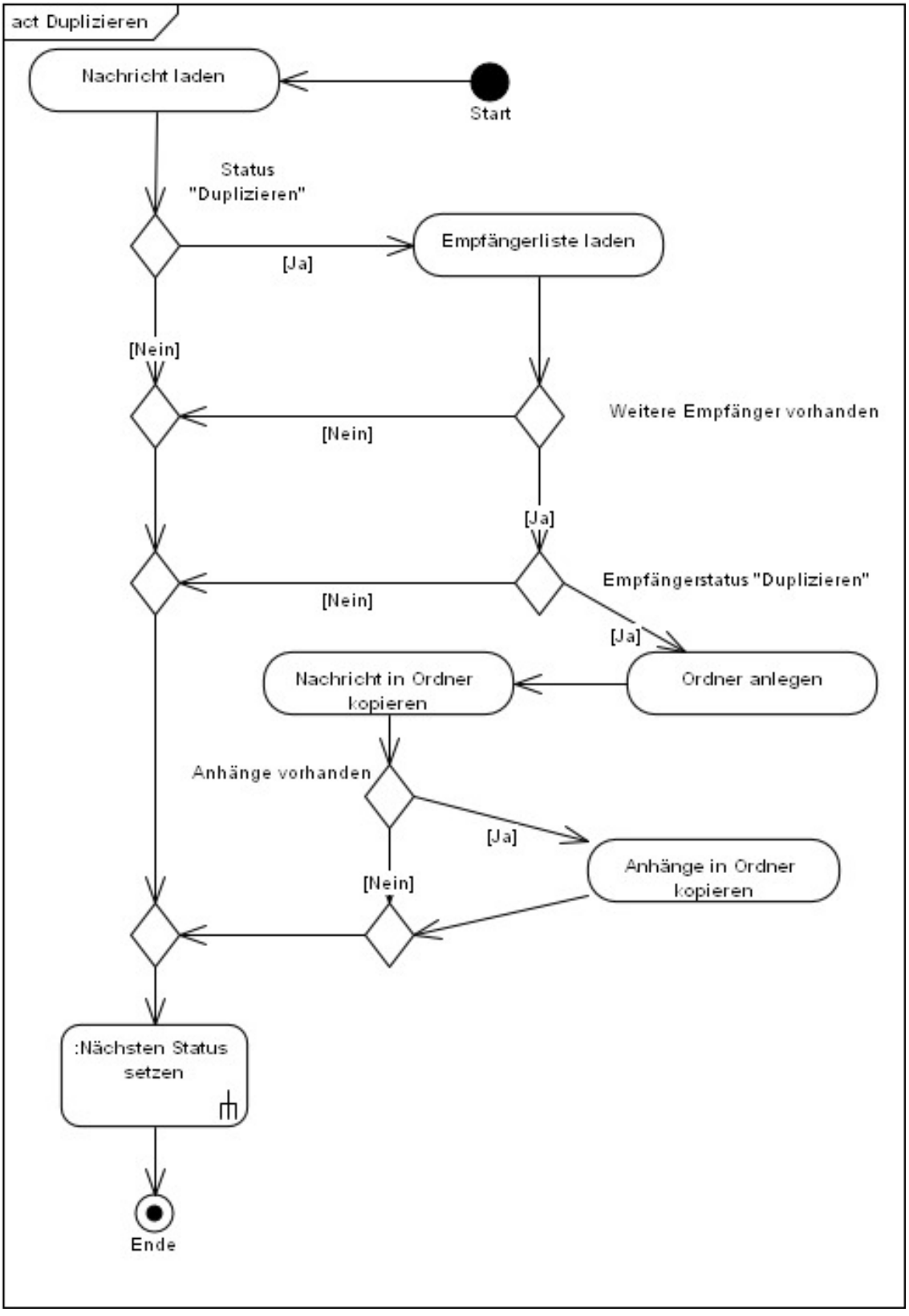

Abbildung 4.18: Systemablauf: Duplizieren

Das Duplizieren der Nachricht auf alle Empfänger dient dazu, dass für jeden Empfänger die Nachricht autonom transformiert und transportiert werden kann. Sollte es einmal zu Engpässen kommen, so kann man die nachfolgenden Schritte auch mittels Multithreading oder Multiprocessing abarbeiten.

4.5.3.5 Transformieren

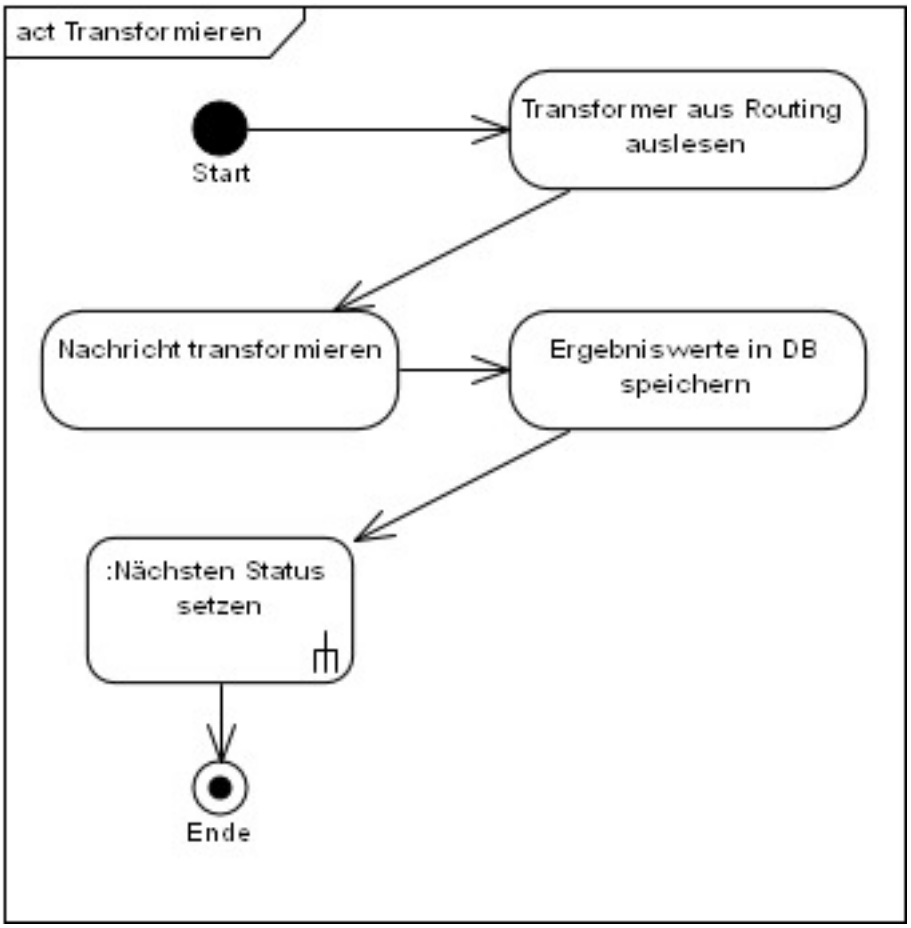

Abbildung 4.19: Systemablauf: Transformieren

Das Transformieren der Nachricht ist der Teil, welcher am aufwendigsten ist, da dieser für jeden Dokumententyp zu geschehen hat.
Vom Ablauf her wird als erstes der Transformator aus dem Routing ausgelesen, die Nachricht transformiert, das Ergebnis in der Datenbank festgehalten (Metadaten wie UUID des Ergebnisses) und zum Abschluss wird der Status auf den nächsten Schritt gesetzt.

4.5.3.6 Transportieren

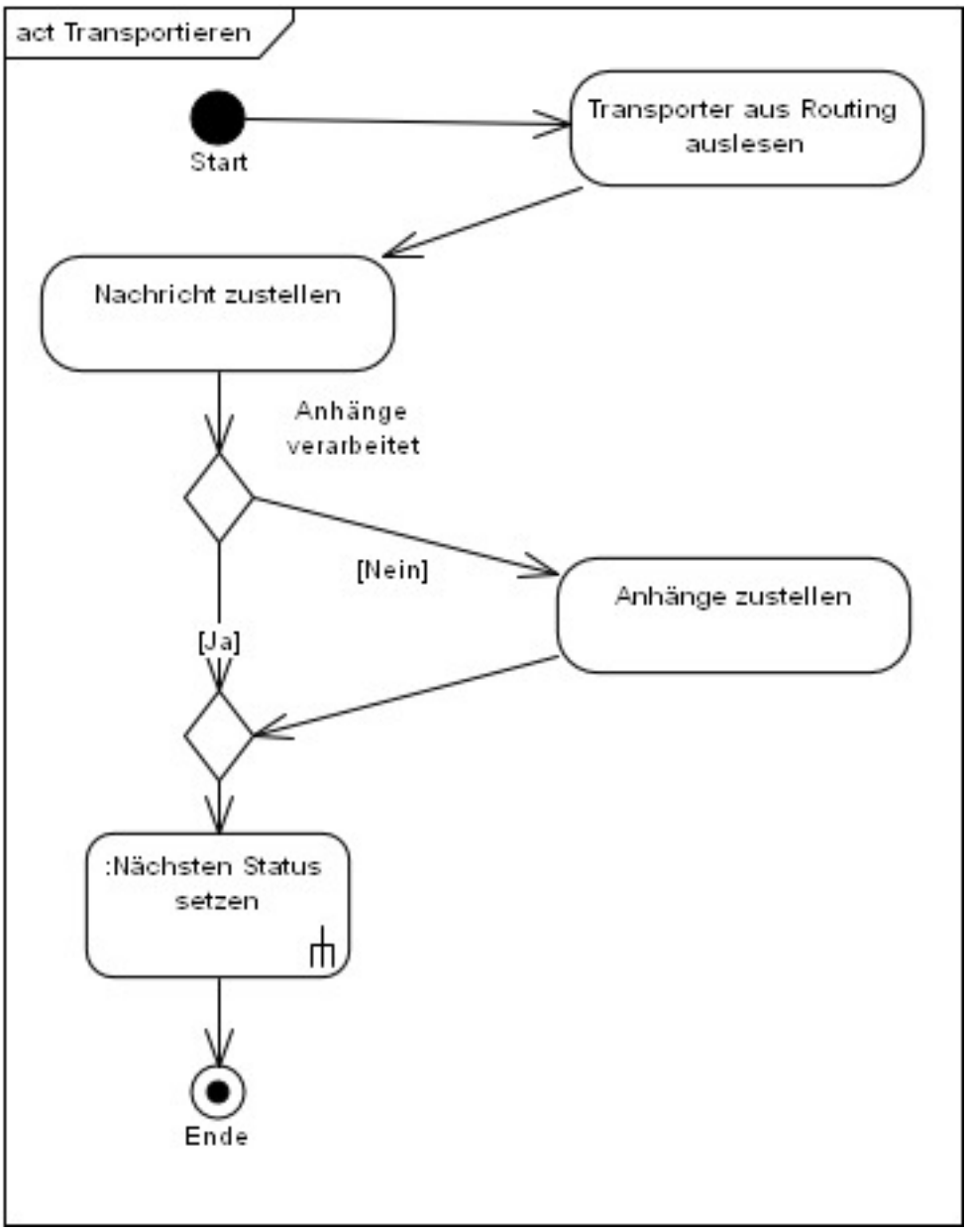

Abbildung 4.20: Systemablauf: Transportieren

Das Transportieren der Nachricht ist der Teil der Nachrichtenverarbeitung, wo die Nachricht dem Empfänger zu gestellt wird. Dazu wird aus dem Routing der Transporter[44] ausgelesen und die Nachricht wird zugestellt. Da nicht alle Formate eine Verarbeitung der Anhänge erlauben[45], müssen die Anhänge, falls es notwendig ist, separat zugestellt werden. Zum Schluss wird noch der Status der Nachricht auf den nächsten Schritt gesetzt, damit die Verarbeitung weiter gehen kann.

[44] Teil der Applikation, welches eine Nachricht transportieren kann.
[45] Im PDF werden die Anhänge eingebettet, in Textdateien werden diese nur referenziert.

4.5.3.7 Abschliessen

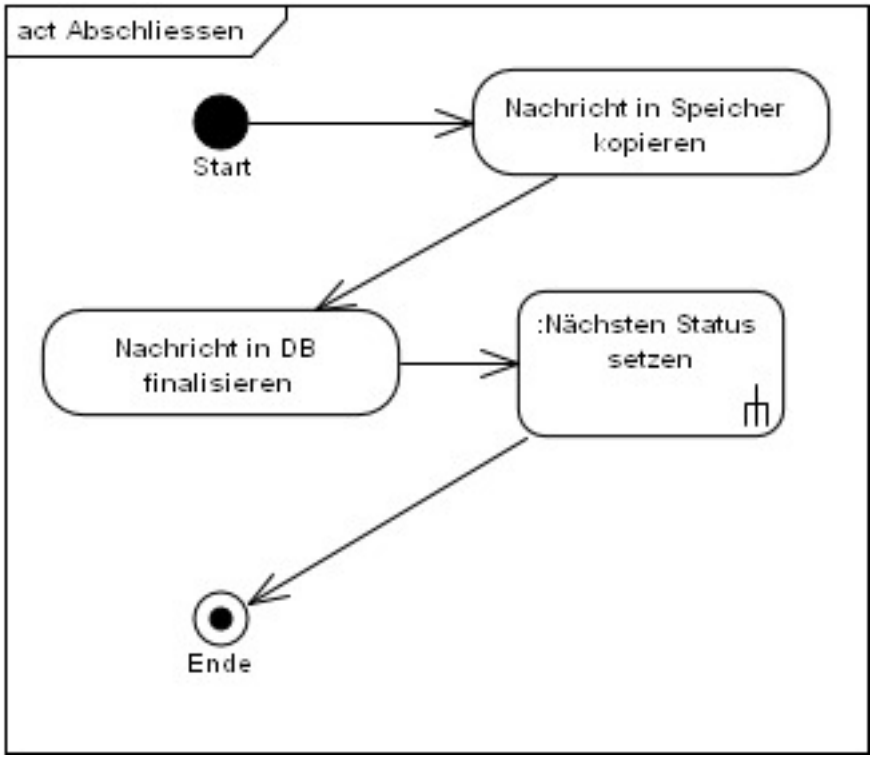

Abbildung 4.21: Systemablauf: Abschliessen

Als letzter Schritt der Verarbeitung wird das erstellte Ergebnis in den Speicher kopiert und die Metadaten der Nachricht in der Datenbank persistiert. Damit ist die Nachricht komplett verarbeitet worden und kann nun zu statistischen Zwecken verwendet werden und/oder der Ablaufverfolgung dienen.

4.5.3.8 Nachricht validieren

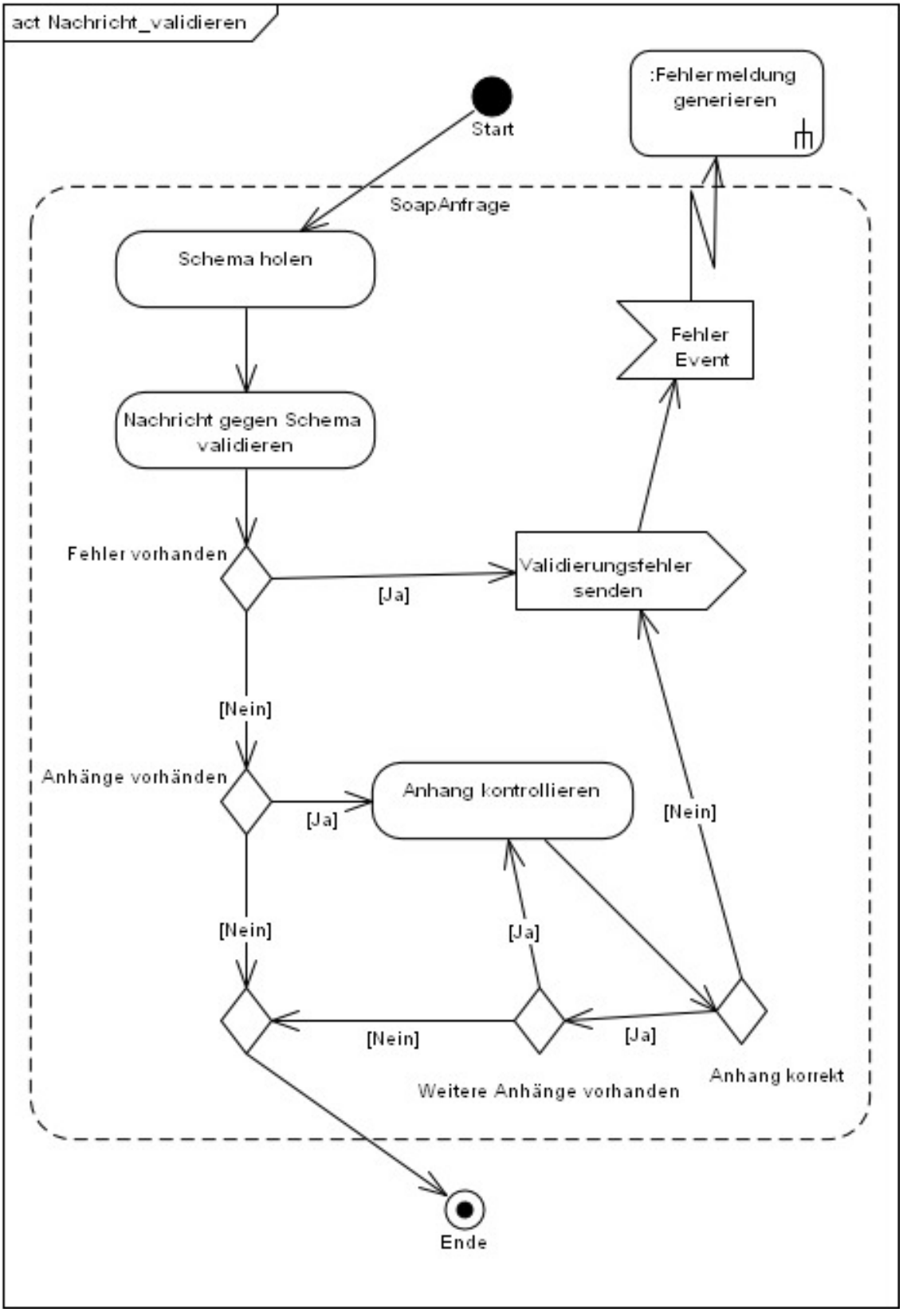

Abbildung 4.22: Systemablauf: Nachricht validieren

Der Validierungsschritt dient dazu, dass nur syntaktisch richtige Nachrichten weiterverarbeitet werden. Dazu wird das dazugehörige Schema geladen und die Nachricht gegen dieses validiert. Sollten hier Fehler vorhanden sein, so wird ein Validierungsfehler erzeugt. Aus diesem Validierungsfehler wird nun eine gültige Fehlermeldung generiert und an den Aufrufer in der Response zurück geschickt.
Sollten auch Anhänge in Nachricht referenziert sein, so wird überprüft, ob diese Anhänge bereits dem System bekannt sind, ansonsten kommt es zu einem Fehler, welcher wiederum dem Aufrufer im Response zurück gegeben wird. Die übergebene Nachricht wird im Fehlerfall nicht dem System zur Verarbeitung übergeben.

4.5.3.9 Nachricht bearbeiten

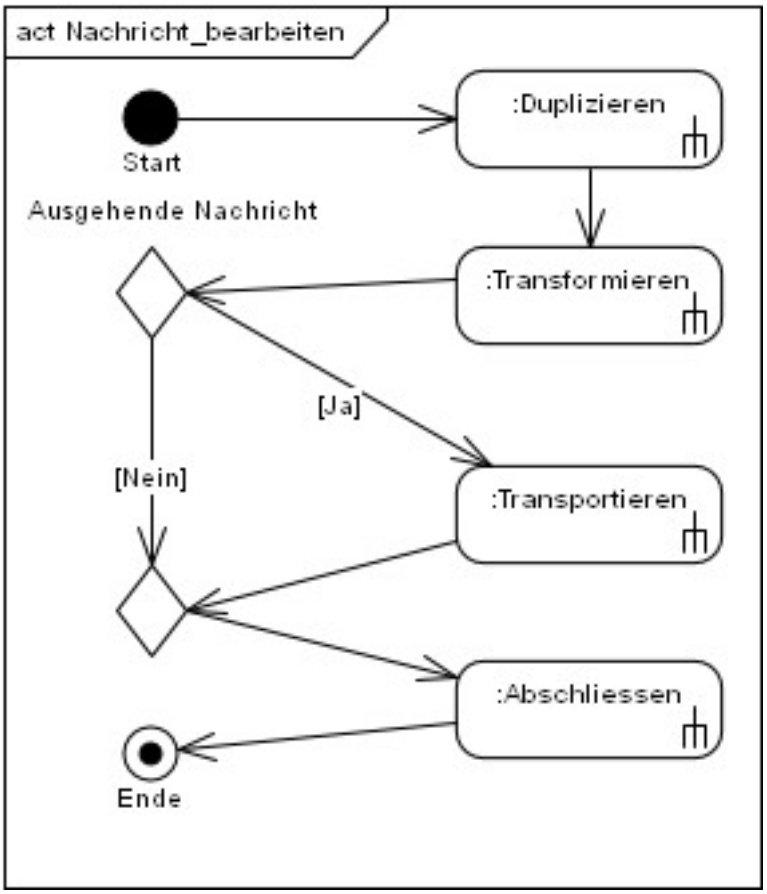

Abbildung 4.23: Systemablauf: Nachricht bearbeiten

Das Bearbeiten einer Nachricht sieht derzeit 4 Schritt vor:

1. das Duplizieren[46]
2. das Transformieren[47]

[46] Siehe dazu 4.5.3.4
[47] Siehe dazu 4.5.3.5

3. das Transportieren[48], wenn es eine Nachricht an das Ausland ist
4. das Abschliessen[49]

Diese Schritte werden für jede Nachricht separat durchlaufen.

[48] Siehe dazu 4.5.3.6
[49] Siehe dazu 4.5.3.7

5 Implementierung

Das NG wird in Java implementiert und soll auf einem JBoss 7 Application Server (JBoss AS) von Red Hat, Inc. (RedHat) laufen. Des weiteren wird eine Datenbank benötigt, wofür hierbei eine Oracle 12c verwendet wird.

5.1 Einrichtung der Entwicklungsumgebung

Dazu benötigt man folgende Elemente:

1. Java SDK (1.7 oder 1.8)[50]
2. Apache Maven (Apache Maven) [51]
3. Oracle Database 12c[52]
4. Oracle JDBC - Treiber
5. IntelliJ[53] als Integrierte Entwicklungsumgebung (IDE)
6. JBoss 7 Application Server[54]

Nachdem man alles von den Seiten auf den Rechner geholt hat, kann man mit em einrichten beginnen.
Zuerst installiert man das JavaSDK. Dann muss man die Systemvariable „JAVA_HOME“ setzen, welche auf das Basisverzeichnis des JavaSDKs zeigt. Es ist auch gut, wenn man den Befehl java bzw. das bin - Verzeichnis in die PATH - Variable einträgt.
Dann kann man Apache Maven installieren und das Apache Maven - Basisverzeichnis in die Systemvariable „M2_HOME“ und das bin - Verzeichnis in die PATH - Variable eintragen.Als nächstes kommt die Oracle Corporation (Oracle) Datenbank an die Reihe. Da das verwendete Betriebssystem Fedora[55] ist das Aufsetzen der Datenbank relativ einfach[56].

[50] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[51] `maven.apache.org` in der Version 3.3.9
[52] `http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html`
[53] `https://www.jetenterprisearchitectswimlanesbrains.com/idea/download/#section=linux`
[54] `http://download.jboss.org/jbossas/7.1/jboss-as-7.1.1.Final/jboss-as-7.1.1.Final.zip`
[55] Downloadbar von `https://getfedora.org/de/`
[56] Informationen zur Installation bekommt man unter `https://oracle-base.com/articles/12c/oracle-db-12cr1-installation-on-fedora-20`

5.1.1 JBoss AS konfigurieren

Der JBoss AS ist einfach in einem Ordner zu entpacken und somit schon einsatzbereit. Damit dieser noch mit der Datenbank kommunizieren kann muss man folgende Schritte tun:

1. Ein Modul anlegen, welches den Datenbanktreiber beinhaltet
2. DataSource und Treiber in der standalone.xml - Datei eintragen

5.1.1.1 JBoss AS Modul anlegen

Dazu benötigt man noch den jdbc - Treiber von Oracle[57]. Dann legt man ein Verzeichnis „ojdbc/main“ im Ordner „modules“ an und kopiert die Datei in den Ordner. Sollte diese anderes als „ojdbc7.jar“ heißen, so ist diese auf „ojdbc7.jar“ um zu benennen, da sonst der weitere Verlauf nicht gewährleistet ist. Des weiteren benötigt man eine Datei, welche module.xml heißt und in Listing 5.1 beschrieben ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<module name="ojdbc" xmlns="urn:jboss:module:1.0">
  <resources>
    <resource-root path="ojdbc7.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

Quelltext 5.1: JBoss Module.xml

5.1.1.2 JBoss AS Datenbank anbinden

Damit der JBoss AS mit der Datenbank kommunizieren kann, muss man nun noch eine Datenbankverbindung, eine DataSource, anlegen. Ein Beispiel für so einen DataSource ist in Listing 5.2 zu sehen, bei der „SID“ durch die ServiceID der Oracle-Instanz zu ersetzen ist, genauso wie die Felder „user-name“ und „password“.

```
<subsystem xmlns="urn:jboss:domain:datasources:1.2">
  <datasources>
    <datasource jndi-name="java:jboss/datasources/NatGatewayDS"
      pool-name="NatGatewayDS" enabled="true" use-ccm="false">
      <connection-url>jdbc:oracle:thin:@localhost:1521:SID</connection-url>
      <driver>oracle</driver>
      <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
      <pool>
```

[57] http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

```
9         <max-pool-size>25</max-pool-size>
10      </pool>
11      <security>
12        <user-name>gateway</user-name>
13        <password>gatewaypass</password>
14      </security>
15      <validation>
16        <check-valid-connection-sql>select 1 from
17          dual</check-valid-connection-sql>
18        <background-validation>true</background-validation>
19        <background-validation-millis>20000</background-validation-millis>
20      </validation>
21    </datasource>
22    <drivers>
23      <driver name="h2" module="com.h2database.h2">
24        <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
25      </driver>
26      <driver name="oracle" module="ojdbc">
27        <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-
              datasource-class>
28      </driver>
29    </drivers>
30  </datasources>
31 </subsystem>
```

Quelltext 5.2: JBoss datasource configuration

5.1.2 Apache Maven Projekt anlegen

Apache Maven ist das Buildtool, welches gewählt wurde um das Projekt zu kompilieren und die zu liefernden Artefakte zu erzeugen. Damit dies sinnvoll funktionieren kann und sich andere Entwickler schnell damit zu recht finden, muss eine Struktur gewählt werden. Diese kann man zwar in einem Refactoringschritt wieder ändern, aber gewisse Rahmenbedingungen sollten schon vorliegen.
Die gewählte Struktur sieht folgendermaßen aus:

- environment
  Laufzeitumgebungen zum testen
  - batch
    Testumgebung für die Stapelverarbeitung
  - data
    Der Datenspeicher für die Verarbeitung
  - jboss
    Der JBoss AS zum testen von einzelnen Funktionen und dem Deployment
- nat-gateway-api
  Hier werden die allgemeinen Komponenten und interne Schnittstellenbeschreibungen

abgelegt

- nat-gateway-batch
  Die Stapelverarbeitung wird als „All-In-One“-jar ausgeliefert und hat hier die Startroutinen hinterlegt
- nat-gateway-delivery
  Auslieferungsprojekt

  - nat-gateway-delivery-batch
    Auslieferungspaket für die Stapelverarbeitung
  - nat-gateway-delivery-doc
    Auslieferungspaket, welches die Dokumentation für die Auslieferungsversion
  - nat-gateway-delivery-sql
    Auslieferungspaket, welches die etwaige Datenbankänderungsscripte beinhaltet, bzw. Scripte, welche Datenmanipulationen in der Datenbank beinhalten
  - nat-gateway-delivery-webservice
    Auslieferungspaket, welches die SOAP-Services enthält
- nat-gateway-persistence
  Modul, in dem die Datenbankanbindung realisiert wird (DAO - Schicht)
- nat-gateway-schemas
  Modul, welches zum Generieren von Javaklassen aus zugelieferten XML-Schema (XSD)-Dateien
- nat-gateway-service
  Das Servicemodul enthält alle Serviceimplementierungen (Business - Schicht)
- nat-gateway-testclient
  Modul mit verschiedenen Testclients zum testen von einzelnen Funktionalitäten, welche intern als auch extern sein können
- nat-gateway-wsclients
  Modul, zur Anbindung von Fremdsystemen mittels SOAP
- nat-gateway-wsdl
  Modul, welches die SOAP-Schnittstelle spezifiziert. Hier werden auch die Klassen zur internen Verwendung generiert
- nat-gateway-ws-ear
  Enterprisearchiv mit allen Abhängigkeiten (dieses Artefakt wird dann am JBoss AS deployed)
- nat-gateway-ws-ejb
  Implementierung der Webserviceschicht

Damit nun die ganzen Module in den Buildprozess eingebunden werden, wird ein „pom.xml“ benötigt, welches im Quelltext 5.3 beschrieben ist.

```
1 <?xml version='1.0' encoding='UTF-8'?>
```

```
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
      /2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM
      /4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>nat.gateway</groupId>
5   <artifactId>nationales-gateway</artifactId>
6   <version>1.0.0-SNAPSHOT</version>
7   <packaging>pom</packaging>
8   <name>Nationales Gateway Maven Root</name>
9   <description>Root pom for project Nationales Gateway</description>
10  <properties>
11    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
13    <maven.build.timestamp.format>dd.MM.yyyy</maven.build.timestamp.format>
14  </properties>
15  <modules>
16    <module>nat-gateway-wsdl</module>
17    <module>nat-gateway-wsclients</module>
18    <module>nat-gateway-schemas</module>
19    <module>nat-gateway-api</module>
20    <module>nat-gateway-persistence</module>
21    <module>nat-gateway-service</module>
22    <module>nat-gateway-batch</module>
23    <module>nat-gateway-testclient</module>
24    <module>nat-gateway-ws-ejb</module>
25    <module>nat-gateway-ws-ear</module>
26    <module>nat-gateway-delivery</module>
27  </modules>
28  <build>
29    <defaultGoal>install</defaultGoal>
30  </build>
31 </project>
```

Quelltext 5.3: Haupt - Pom des Projektes

5.2 Softwarearchitektur

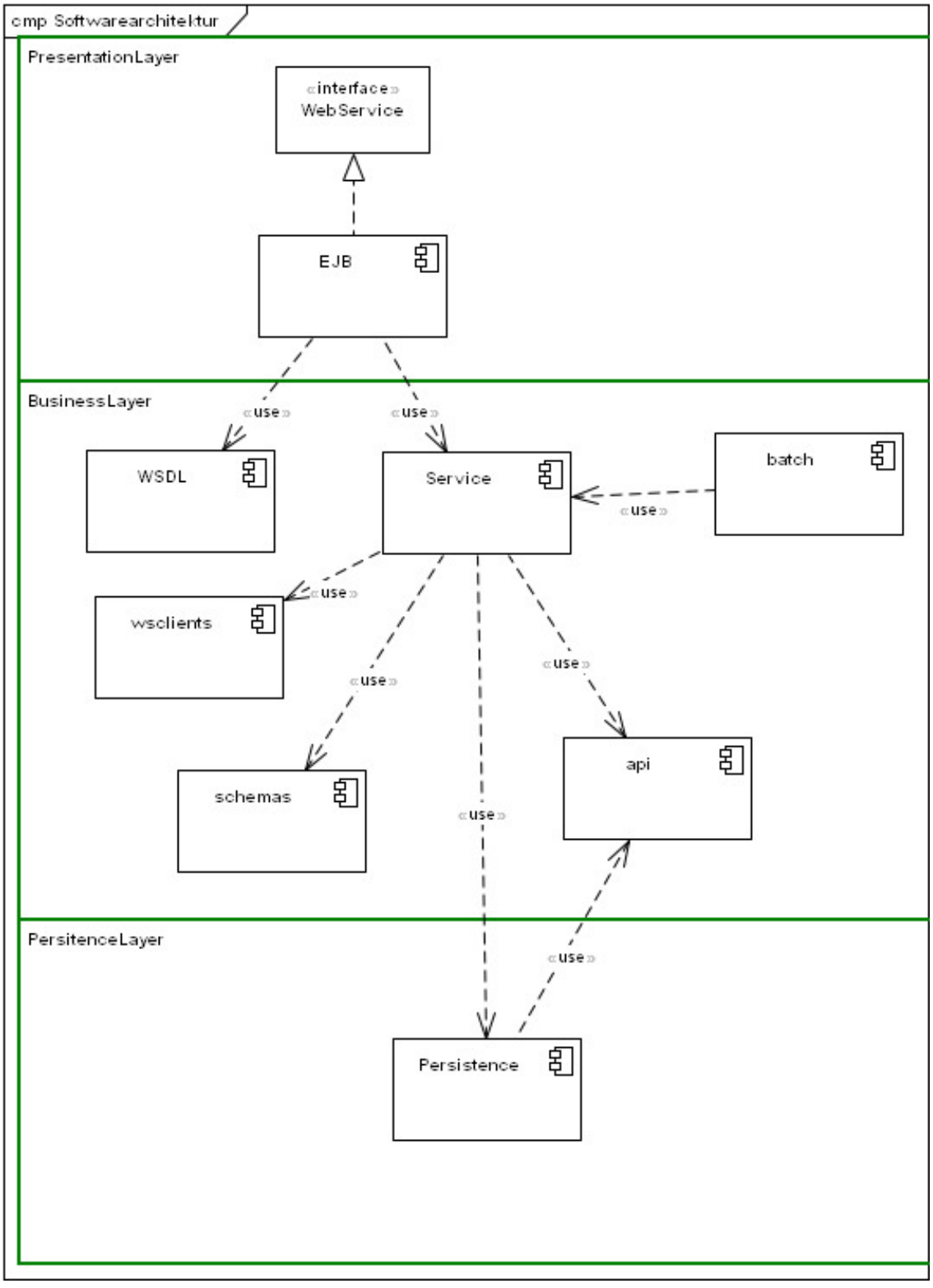

Abbildung 5.1: 3 - Schichtarchitektur

Die Architektur des Softwareprojektes ist auf der 3-Schichtarchitektur aufgebaut. D.h., dass es eine Trennung zwischen den Funktionalitäten und den Einsatz der Module gibt. In Abbildung 5.1 sind die einzelnen Module auf die Schichten verteilt, damit man einen Überblick bekommt, welche Aufgaben jedes Modul hat.

5.2.1 Persistence-Layer

Die Persistence-Schicht oder auch Datenschicht kümmert sich um den Zugriff auf die Daten in physischer Form. Das bedeutet, dass hier die Zugriffe auf die Datenbank und auf das Dateisystem abgehandelt werden. Damit das System modular bleiben kann, ist dies die einzige Schicht, welche auf die Datenspeicher zugreift.

5.2.2 Business-Layer

Die Business-Logic-Schicht oder auch Verarbeitungsschicht, beinhaltet alle logischen Abläufe, damit das System funktionieren kann.

5.2.3 Presentation-Layer

Der Presentation-Layer oder auch Anzeigeschicht veröffentlicht die Resultate des Business-Layers. In unserem Fall werden die Anfragen der Backend-Systeme von dem SOAP-WebService entgegengenommen, der ServiceSchicht zur Verfügung gestellt und über diese ebenso abgeholt.

5.3 Beschreibung der Module

In diesem Abschnitt werden die wichtigsten Module und deren Zweck beschrieben. Wie Module untereinander agieren, ergibt sich aus der Abbildung 5.1.

5.3.1 nat-gateway-wsdl

Dieses Modul beinhaltet die SOAP-Schnittstellenbeschreibung, welche in der Form eines Web Services Description Language (WSDL)s vorliegen muss, damit nach dem „Contract-First“ Prinzip vorgegangen werden kann. Die verwendeten WSDL-Dateien und die dahinter liegenden Datenbeschreibungen können im Anhang B.1 nachgelesen werden. Innerhalb dieses Moduls werden die WSDL-Dateien verarbeitet und mittels dem Apache Maven-Plugin „cxf-codegen-plugin“ von Apache CXF (CXF) werden aus WSDL-Dateien Java-Klassen generiert. Diese Klassen bilden die Grundlage für die Schnittstellen und die weitere Vorgehensweise.

5.3.2 nat-gateway-api

Das API - Modul stellt die Kommunikationsfähigkeit innerhalb der Software mit den anderen Modulen sicher. Es beinhaltet alle Klassen, welche zwischen den Paketen ausgetauscht werden. Weiteres werden hier auch verschiedene Hilfsklassen implementiert, welche allen Modulen zur Verfügung gestellt werden. Hier werden auch die unterschiedlichen DTOs[58] implementiert.

5.3.3 nat-gateway-persistence

Die Persistence-Schicht stellt den Zugriff auf die Datenbank sicher. Damit nicht jedes Modul von der Datenbank erfahren muss, werden hier die DAO-Objekte[59] implementiert. Damit ist alles, was die Datenbank betrifft in einem Modul zusammengefasst.

5.3.4 nat-gateway-service

Das Servicemodul ist das größte der Module und enthält die ganze Logik. Die grundlegende Idee dabei ist, dass jedes Service ein Interface[60] implementiert, damit man das Service gegebenenfalls tauschen kann. Damit ist gewährleistet, dass man auch an andere Implementierungen testen kann, ohne gleich alles neu zu schreiben.

5.3.5 nat-gateway-ws-ejb

Innerhalb des EnterpriseJavaBean (EJB)-Moduls befinden sich alle Elemente der Webservices. Dieses Modul wird als JAR[61] geliefert.

5.3.6 nat-gateway-ws-ear

Im EAR[62]-Modul wird das Paket für den Applikationsserver zusammengebaut. Dies ist vorsorglich schon gemacht worden, damit man nachher auch noch eine theoretische

[58] Data Transfer Object (DTO) - Dienen der Kommunikation zwischen den Schichten. Sind die auszutauschenden Daten definiert, so kann man die dahinter liegenden Services leichter austauschen. Der Nachteil hinter den DTOs, ist der zusätzlichen Aufwand des Erzeugens dieser. Also Modularität vs. Geschwindigkeit

[59] Data Access Object (DAO) - Bilden eine Schicht zwischen der Software und den Datenbanken. Damit werden die Zugriffe auf die Datenbank kanalisiert und die Software kann bei Bedarf die Zugriffsart tauschen. Z.B. von Datenbank auf Dateisystem

[60] Ein Interface ist eine Javaklasse, welche nur die Methodenrümpfe enthält, ähnlich wie ein Header-File in C++

[61] Das JAR (Java ARchiv) ist die Basis für jedes Javaprogramm.

[62] Das EAR (Enterprise ARchiv) dient der Auslieferung von mehreren Applikationen mit gleichem Hintergrund.

Administrationsoberfläche oder andere Oberflächen mit ausliefern kann ohne die Auslieferungsregeln ändern zu müssen. Ansonsten würden die Webservices in einem WAR[63] ausgeliefert.

5.3.7 nat-gateway-batch

Das Batch-Modul dient der Stapelverarbeitung als Grundlage für den Betrieb. Hier sind die Startroutinen und die damit verbundene Fehlerbehandlung enthalten.

5.4 Implementierungsvorgang

Der Implementierungsvorgang teilt sich in 2 Abschnitte, zum einen ist dies die Implementierung der Webserviceschnittstelle, welche der Datenannahme und der Datenabgabe dient und zum anderen die Implementierung der Stapelverarbeitung, welche das Verarbeiten der Nachrichten garantieren soll.

5.4.1 Implementierung der Webserviceschnittstelle

Die Implementierung erfolgt nach dem „Contract-First“ Prinzip, wobei die Schnittstellendefinition das erste Element ist. Das bedeutet, dass das Schreiben der WSDLs der erste Schritt ist. Dazu werden die Datentypen, welche im Anhang unter A.1 beschrieben sind, herangezogen. Daraus werden die einzelnen Schnittstellendefinitionen geschrieben, welche im Anhang unter B.1 zu finden sind. Damit nicht immer die einzelnen Datentypen dupliziert werden müssen, werden diese in eine eigene XSD-Datei ausgelagert. Damit werden die Datentypen wiederverwendbar und überschreiben sich nicht beim Generierungsvorgang, da diese alle im gleichen Namespace[64] liegen.
Um die Webservices auch aufrufen zu können, werden die generierten Klassen in eine EJB-Umgebung eingebunden, damit die Webservices von JBoss AS auch zur Verfügung gestellt werden können. Damit ein EJB als Webservice zur Verfügung gestellt werden kann, benötigt man Definitionen über der Klassendeklaration, welche im Quelltext 5.4 angegeben sind. Dabei ist die Zeile 1 die Aufforderung, dass diese Klasse als EJB einzusetzen ist und die Zeile 4 ist die Aufforderung, dass diese Klasse als Webservice zur Verfügung gestellt werden soll. Wichtig dabei ist, dass nur zustandslose EJBs als Webservice angeboten werden können, da SOAP per Definition zustandslos ist. Die Parameter der Zeilen 5 bis 9 dienen der genaueren Spezifizierung des Webservices, wobei die Zeile 8 auf das selbst geschriebene WSDL verweist. Würde diese Zeile fehlen, muss der Container das WSDL erzeugen, was dem „Code-First“ Prinzip entsprechen würde und dieses zum Download bereit stellen. Die Zeile 2 weist den Container

[63] Das WAR (Web ARchiv) ist eine ZIP-Datei, welche in der Regel eine Webapplikation enthält.
[64] Der Namespace bildet den Namensraum für Datentypen innerhalb von XML-Dateien. Dieser Namespace wird auch bei der Generierung herangezogen, um diese in Java-Packages zu legen.

an, dass binäre Daten mittels MTOM[65] versendet und empfangen werden. Die Zeile 3 besagt, dass die Nachrichten nach dem SOAP-Protokoll in der Version 1.2 gesendet und empfangen werden.

```
1 @Stateless(name = "InformationEJB")
2 @MTOM
3 @BindingType(SOAPBinding.SOAP12HTTP_BINDING)
4 @WebService(
5     endpointInterface = "nat.gateway.messages.InformationServicePort",
6     name = "InformationService", //PortName
7     serviceName = "InformationService", //in wsdl = service:name
8     wsdlLocation = "/META-INF/wsdl/InformationService.wsdl",
9     targetNamespace = "http://gateway.nat/messages/"
10 )
```

Quelltext 5.4: EJB als WebService

Nachdem die Webservices nun lieferbar sind und angesprochen werden können, kommt nun die Serviceschicht an die Reihe. Hierbei werden die Services im ersten Schritt sehr rudimentär angelegt und eine Standardimplementierung verleiht dem ganzen einen Hauch von Funktionalität. Hier werden auch schon die ersten logischen Abläufe abgebildet,wie z.B. der Validierungsschritt, welcher im Abschnitt 4.5.3.8 beschrieben ist. Wenn die grundlegenden Funktionen der Serviceschicht bekannt sind, kann man sich auch an das Implementieren der Datenhaltungsschicht machen und einmal die Basisservices[66] implementieren.
Der nächste Schritt ist nun der Entwurf der Datenbankarchitektur. Hierbei muss nun auf die Anforderungen aus Kapitel 4.2 geachtet werden, insbesondere auf ANF018. Aus der Anforderung ANF018 kann man ableiten, dass es sinnvoll ist, eine Datenbankstruktur mit 2 Strukturen zu haben. Ein Teil davon sind die Arbeitsdaten und der zweite Teil sind dauerhaft gespeicherte Daten. Des weiteren ergibt sich aus den Anforderungen ANF012 und ANF013, dass es ein 1:n Beziehung zwischen einer Nachricht und den Empfängern geben muss. Daraus ergibt sich die Datenbankstruktur, welche in Abbildung 5.2 vorliegt. Hierbei stellt die Tabelle *MessageWork* die ankommende Nachricht dar. Die Tabelle *BatchWork* gibt Auskunft über den Verarbeitungszustand der Nachricht und die Tabelle *RecMulticast*[67] stellt die Empfänger einer Nachricht dar. Diese drei Tabellen fungieren als Arbeitstabellen, d.h., dass die darin enthaltenen Daten nur solange verfügbar sind, solange diese benötigt werden. Wenn die Nachricht abgearbeitet ist, können die Einträge gelöscht werden. Dieser Vorgang wird im Kapitel 4.5.2.3 wiederum definiert. Die Tabelle *Message* ist das Ergebnis der Bearbeitung einer Nachricht und muss alle Metadaten der ursprünglichen Nachricht, als auch der konvertierten und

[65] Message Transmission Optimization Mechanism (MTOM) bezeichnet die Methode, wie binäre Daten übertragen werden. Dabei werden diese nicht als Base64-codierte Character DATA (CDTA)-Blöcke gesendet, sondern als Referenzen, welche am Schluss an die Nachricht angehängt sind.

[66] Die Basisservices der Datenhaltungsschicht bezeichnet man auch als Create, Read, Update and Delete (CRUD)-Services.

[67] RecMulticast setzt sich aus den Worten Receiver (Empfänger) und Multicast(Gruppenzustellung) zusammen.

zugestellten Nachricht enthalten. Der dahinter liegende Vorgang wird in Kapitel 4.5.3.7 beschrieben.

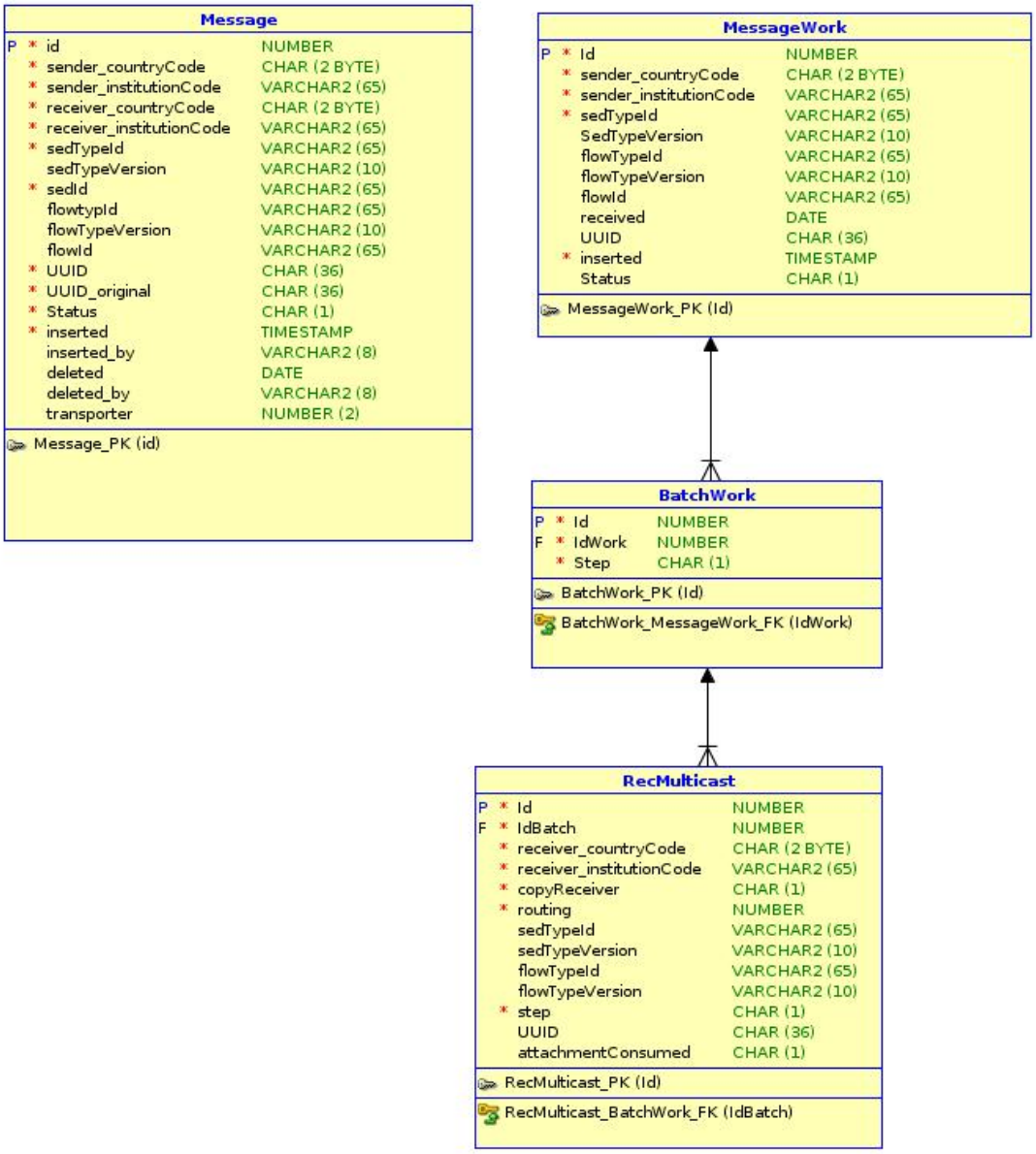

Abbildung 5.2: Datenbankstruktur für die Nachrichten

5.4.2 Implementierung der Stapelverarbeitung

Die Stapelverarbeitung hat 3 UCs, welche auch in der Reihenfolge der Nummerierung durchlaufen werden sollen. Dafür werden in der Startroutine genau diese 3 Punkte durchlaufen. Das Snippet im Quellcode 5.5 zeigt, wie dieser Vorgang implementiert ist, wobei die Variable *failure* eine boolsche Variable ist, welche den Gesamtstatus der Stapelverarbeitung aufnimmt und diesen dann an den Aufrufer zurück liefert.
In den Zeilen 2 bis 5 wird die Liste der möglichen Eingabesysteme nach neuen Nach-

richten durchsucht und in die Datenbank eingetragen. Die Zeile 6 macht nun die ganze Arbeit und bearbeitet die Nachrichten. Der dahinter liegende, Ablauf ist im Kapitel 4.5.3.9 beschrieben. Die Zeilen 7 bis 10 räumen zum Schuss das System auf und sorgen dafür, dass der Datenverbrauch geringer bleibt.

```
1   boolean failure = false;
2   for (InputSearch inputSearch : inputSearchList) {
3     log.info("run() - Starting import with {}", inputSearch.name());
4     failure = inputSearch.importingFromSource() || failure;
5   }
6   failure = getWorker().work() || failure;
7   for (Cleaner cleaner : cleaners) {
8     final int clean = cleaner.clean();
9     cleaned += clean;
10  }
```

Quelltext 5.5: Ablauf der Stapelverarbeitung

5.4.3 Implementierung der Transformation in ein PDF

Wenn man ein PDF erzeugen möchte, muss man sich zuerst die Quelldaten ansehen, um im Anschluss ein geeignetes Werkzeug dafür zu finden. Derzeit sind mir 2 Werkzeugarten für JAVA bekannt:

- iText
  Eine proprietäre Software **iText Group NV**, welche mittels JAVA-API zu verwenden ist
- Apache-FOP
  Apache FOP (FOP) ist ein Transformator, welcher aus einer fo-Datei ein PDF erzeugt.

Damit man in der Handhabung der Software flexibel bleibt, wird für die PDF-Transformation FOP gewählt. Dabei liegt die Stärke nicht darin, dass aus einer fo-Datei ein PDF erzeugt wird, sondern darin, dass man aus einem XML mittels eXtensible Stylesheet Language - Formatting Objects (XSL-FO) eine fo-Datei erzeugt und diese direkt an den Prozessor übergeben kann.

5.4.3.1 Apache FOP implementieren

Die Verwendung des Prozessors ist relativ einfach und wird in dem Codesnippet 5.6 beschrieben. Der Ablauf dabei ist, dass man einen *Transformer* mit einem eXtensible Stylesheet Language (XSL) erzeugt (Zeilen 1 bis 3) und diesem eine Quelle (Zeile 4) und ein Resultat (Zeilen 5 und 6) übergibt (Zeile 7). Das ist der ganze theoretische Aufwand. Damit kann man sich schon an das Schreibern der XSL-FO-Dateien, welche dem *Transformer* als Transformierungsregeln dienen, wenden.

```
1 TransformerFactory tFactory = TransformerFactory.newInstance();
2 Templates cachedXSLT = tFactory.newTemplates(new StreamSource(xslFile));
3 Transformer transformer = cachedXSLT.newTransformer();
4 SAXSource transformSource = new SAXSource(new InputSource(new FileInputStream("
      Datei")));
5 Fop fop = getFop(new FileOutputStream("Ausgabe"));
6 Result res = new SAXResult(fop.getDefaultHandler());
7 transformer.transform(transformSource, res);
```

Quelltext 5.6: Aufruf der Transformation mittels FOP

5.4.3.2 Erstellen der XSL-FO - Dateien

Der Aufbau der XSL-FO-Dateien folgt 3 Prinzipien:

1. Formatierung eines XML-Knotens
2. Aufruf von template-Elementen
3. Auswahl von XML-Knoten mittels XML Path Language (XPath)

Da FOP mit fo-Dateien arbeitet und diese eine XML-Struktur besitzen, muss man die Zeilen aus dem Listing 5.7 verwenden, damit aus der Eingabedatei auch die richtige Ausgabe wird. Die Ausgabe wird dabei mittels der Zeile 2 festgelegt. Des weiteren wird die Version 2.0 von XSL-FO verwendet, welche den Vorteil gegenüber Version 1.0 hat, dass es build-in Funktionen gibt, wie z.B. das heutige Datum einfügen, oder auch einen Text oder Zahlen formatieren.[68]

```
1 <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xs="http://www.w3.org
      /2001/XMLSchema" exclude-result-prefixes="xsl sed cla_sed egda_sed_basic" >
2 <xsl:output method="xml" encoding="UTF-8" indent="yes" standalone="yes"/>
3 ...
4 </xsl:stylesheet>
```

Quelltext 5.7: XSL Header

Dann kommt die wichtigste Zeile. Im Codesnippet 5.8 beginnt nun die Transformation mit der Selektion des root-Knotens. In der Zeile 2 wird nun der root-Knoten der Ausgabedatei geschrieben und zwar *<fo:root*. Damit beginnt die fo-Datei und kann nun mit den gültigen Elementen gefüllt werden. Da die generierten Seiten aller verwendeten Stylesheets immer gleich ist, wird hier die template-Methode pageLayout aufgerufen, welche die Seitengröße und die Seitenreihenfolge definiert. Des weiteren sind die Nutzdaten immer in 2 Teile geteilt. Zum ersten ein globaler Block und der zweite Teil ist eine mögliche Liste an individuellen Elementen. Damit die Verarbeitung immer gleich funktioniert,

[68] Für eine Einführung zum Thema XSL vergleiche [w3s] und eine gute Einführung zum Thema XSL-FO vergleiche [PK]

wurden 2 Templates geschrieben, mit denen man die Verarbeitung anstößt. Dies sind die Templates *makeGlobal*[69] und *makeIndividual*[70] .

```
1 <xsl:template match="/sed:sed">
2   <fo:root font-size="10pt" font-family="Times" font-weight="normal" font-style=
        "normal" letter-spacing="0.1pt">
3     <xsl:call-template name="page-layout"/>
4     <xsl:for-each select="cla_sed:payload/cla_sed:global">
5       <xsl:call-template name="makeGlobal" />
6     </xsl:for-each>
7     <xsl:for-each select="cla_sed:payload/cla_sed:individual">
8       <xsl:call-template name="makeIndividual" />
9     </xsl:for-each>
10  </fo:root>
11 </xsl:template>
```

Quelltext 5.8: Grundlagen XSL

```
1 <xsl:template match="cla_sed:global">
2 ...
3 </xsl:template>
```

Quelltext 5.9: Definition Template makeGlobal

```
1 <xsl:template match="cla_sed:individual">
2 ...
3 </xsl:template>
```

Quelltext 5.10: Definition Template makeIndividual

Nachdem beim Aufruf der Methoden *makeGlobal* und *makeIndividual* bereits der gewünscht Knoten selektiert ist, muss man nun noch den Aufruf aus dem Codesnippet 5.11 einsetzen und es kann der jeweilige Knoten individuell verarbeitet werden.

```
1 <xsl:apply-templates select="."/>
```

Quelltext 5.11: Apply template

5.4.4 Implementierung der Transformation in ein E125 bzw. aus einem E125

Das E125-Formular ist ein Formular mit fixen Längen, d.h., dass in der gelieferten bzw. geschriebenen Datei jeder Position eine eindeutige Interpretierung zu geteilt ist. Das Verfahren des Auslesen aus so einer Datei hat einen generellen Charakter. In Java übergibt man einer Javabean einfach die ganze Zeile und holt sich anhand der Position die einzelnen Werte heraus.
Damit diese Methodik auch für andere Formulartypen verwendbar ist, wurden die Lese-

[69] Siehe Zeile 4 bis 6 und die Definition der Methode in Codesnippet 5.9
[70] Siehe Zeile 7 bis 9 und die Definition der Methode in Codesnippet 5.10

und Schreibemethoden in einer abstrakten Klasse zusammengefasst. Die verwendete Javabean enthält somit nur mehr den Konstruktor und eine Reihe von Feldern, welche mit Feldposition, Position in der Zeile, Länge des Datenfeldes und dem zu erwartendem Datentyp ausgestattet sind. Damit kann man nun sehr einfach die Werte auslesen. Das Schreiben ist auf der gleichen Methodik aufgesetzt, nur dass man nicht alle Werte setzen muss, sondern es werden nur die benötigten Werte gesetzt und die Ausgabe, welche sowieso alle Werte schreiben muss, entscheidet, was, wie geschrieben wird. Hat ein Feld keinen Wert, so werden der Definition nach numerische Werte rechtsbündig geschrieben und mit 0 aufgefüllt und alphanumerische Werte linksbündig geschrieben und mit Leerzeichen erweitert. Im Codesnippet 5.12 sieht man den strukturellen Aufbau so einer Javaimplementierung ohne die ganzen Hilfsmethoden. Wichtig ist in diesem Fall, das Feld mit der Positionsnummer 0, da alle Positionsbeschreibungen mit 1 beginnen. Somit kann man immer die richtige Positionsnummer verwenden. Beim Vorgang des Auslesens, muss man allerdings die Zeilenposition um 1 reduzieren, da in Java die Reihennummerierung mit 0 beginnt und nicht mit 1.

```
1  public class E125Bean extends E1XXBean {
2    private final E1XXBean.Field[] fields = {
3      new E1XXBean.Field(0, 0, 0, String.class),
4      new E1XXBean.Field(1, 1, 10, String.class),
5      new E1XXBean.Field(2, 11, 10, String.class),
6  ...
7      new E1XXBean.Field(69, 1257, 2, String.class),
8      new E1XXBean.Field(70, 1259, 20, String.class),
9      new E1XXBean.Field(71, 1279, 30, String.class),
10   };
11   public E125Bean(String line) {
12     super(line);
13   }
14 }
```

Quelltext 5.12: Fixlängen Javabean

Damit man ein Gefühl dafür bekommt, wie komplex der Aufbau so einer Klasse ist, zeigt Abbildung 5.3 den Zusammenhang, der dafür verwendeten Klassen. Der komplexere Aufbau gewährleistet dabei die Wiederverwendbarkeit, für den Fall, dass es noch weitere Fixlängenformate verwendet werden soll.

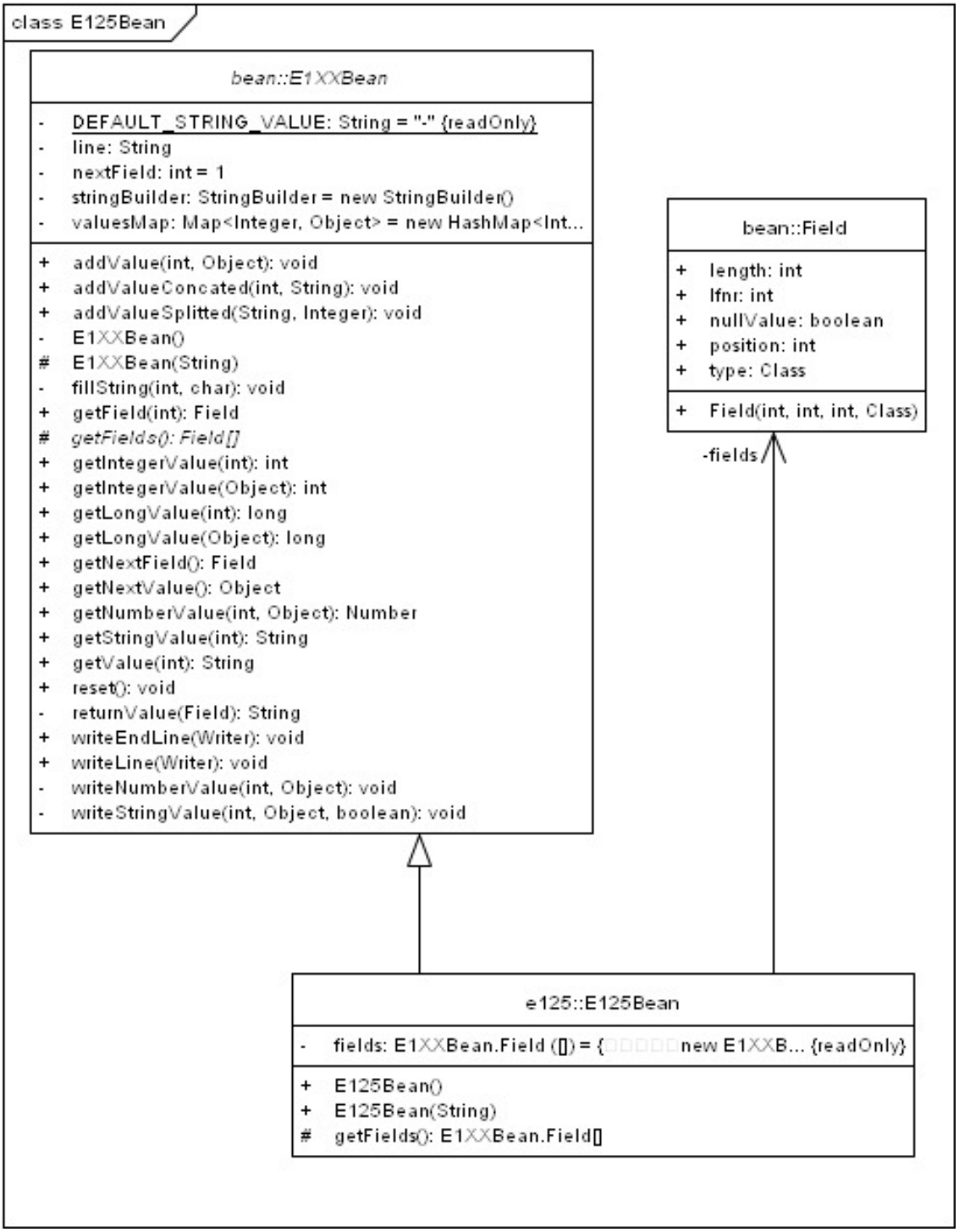

Abbildung 5.3: Ableitungshierarchie des E125Bean's

Für das bessere Verständnis sind hier noch einige Snippets, welche die Funktionalität dieser Javabean veranschaulichen. Als erster Punkt sind die Instanzvariablen zu erwähnen, welche dieser Bean zu eigen sind. Im Codesnippet 5.13 sind die Deklarationen der Bean veranschaulicht. Hier bei sieht man, dass die Bean nur aus der Eingabezeile[71], ei-

[71] Codezeile 3

ner Builderinstanz[72] für die Ausgabe und einer Map[73] besteht. Die restlichen benötigten Dateninformationen werden in der konkreten Implementierung[74] geliefert.

```
1  private static final String DEFAULT_STRING_VALUE = "-";
2  private int nextField = 1;
3  private @Getter String line;
4  private StringBuilder stringBuilder = new StringBuilder();
5  private Map<Integer, Object> valuesMap = new HashMap<Integer, Object>();
```

Quelltext 5.13: Instanzvariablen E1XXBean

Die wichtigste Funkion der E1XXBean ist im Codesnippet 5.14 beschrieben. Diese Funktionalität muss von der konkreten Implementierung zur Verfügung gestellt werden und bezweckt, dass die einzelnen Feldbeschreibungen übergeben werden.

```
1  protected abstract Field[] getFields();
```

Quelltext 5.14: Wichtigste abstrakte Methode

Als letzten wichtigen Aspekt wird noch das Schreiben der Bean[75] vorgestellt. Hierbei wird über die Felder (Fields) iteriert und aus der Map der Wert mit dem Feldindex geholt[76]. Dann wird der Datentyp überprüft und der Wert an eine dementsprechende Schreibmethode übergeben[77]. Zum Schluss werden die einzelnen Zeichen aus dem Builder an einen Writer[78] übergeben und ein Zeilenumbruch angefügt.

```
1  public void writeLine(Writer writer) throws IOException {
2    for (int i = 0; i < getFields().length; i++) {
3      Field field = getFields()[i];
4      Object value = valuesMap.get(field.getLfnr());
5      if (field.getType() == String.class) {
6        writeStringValue(i, value, true);
7      } else if (field.getType() == Long.class || field.getType() == Integer.
           class) {
8        writeNumberValue(i, value);
9      }
10   }
11   for (int i = 0; i < stringBuilder.length(); i++) {
12     writer.write(stringBuilder.charAt(i));
13   }
14   writer.write("\n");
15 }
```

Quelltext 5.15: Schreiben einer Fixlängenzeile

[72] Codezeile 4
[73] Codezeile 5. Eine Map besteht aus einem Key-Value - Paar
[74] Vergleiche Codesnippet 5.12
[75] Vgl. Codesnippet 5.15
[76] Codezeile 4
[77] Codezeilen 6 und 8
[78] Ein Writer ist eine Klasse, welche zum Schreiben an ein darunterliegendes Medium dient. Das Medium kann nun eine Datei auf der Festplatte oder auch die Standardausgabe bei Konsolenapplikationen sein.

6 Kontrolle

Damit man bewerten kann, was erreicht wurde, muss man sich die Frage stellen, was gefragt war.
Die Aufgabenstellung war, dass für ein nationales Gateway eine Analyse im Sinn der Anforderungsanalyse durchzuführen ist und die gewonnenen Erkenntnisse in einer Javaapplikation umzusetzen sind.
Die Analyse hat gezeigt, dass die einzelnen Schritte zur Umsetzung, doch komplexer werden, je genauer man die Vorgänge betrachtet. Das Projektdreieck, welches in Abbildung 2.1 dargestellt ist, beschreibt diesen Sachverhalt sehr treffend.

6.1 Was wurde erreicht

6.1.1 Analyse

Die Analyse wurde soweit abgeschlossen, dass man aus den erlangten Erkenntnissen die Software implementieren kann und diese auch gewissen Standards entspricht. Es wäre allerdings von Vorteil, wenn einige Punkte noch weiter erschlossen worden wären. Wie zum Beispiel, wie die interne Struktur der Applikation aussehen soll, Stichwort Klassendiagramm. Als anderen Punkt ist hier auch zu erwähnen, dass diese Arbeit ausschließlich von mir erstellt wurde und somit ich als alleiniger Stakeholder meine Wünsche und Ideen in die Analysephase eingeflossen sind. In der Regel werden solche Analysen von einer Person moderiert, geleitet und dokumentiert und innerhalb eines Personenkreises erarbeitet. Aus diesem Grund können z.B. die Anforderungen aus dem Kapitel 4.2 nur einen Teil der möglichen Anforderungen abdecken.

6.1.2 Implementierung

Die Implementierung der Software ist im großen und ganzen abgeschlossen und betriebsbereit. Sie ist individuell erweiterbar und die einzelnen Komponenten sind sauber getrennt. Einzig der Punkt der Validierung ist noch zu hinterfragen. Das Problem beim Validieren der Anfragen stellt sich folgendermaßen dar:

6.1.2.1 Validierung mittels CXF-Interceptor

Der CXF-Interceptor kann eine SOAP-Anfrage oder -Antwort annehmen und diese gegen das WSDL validieren. Dabei gibt es 2 Szenarien, die zu beachten sind:

- Anfrage mit geringen Datenvolumen
  Die SOAP-Anfrage wird dem Validator übergeben und diese wird validiert.
- Anfrage mit hohem Datenvolumen
  In diesem Fall kann es zu Problemen kommen, da die Nutzdaten in Form eines binären Anhangs übertragen werden. Dieser Anhang wird zwar nicht validiert, aber dieses Datenpaket wird in ein Bytearray konvertiert und benötigt somit Speicher, welcher aus dem HEAP entnommen wird. Daraus ergibt sich, dass die maximal zu übertragende Paketgröße mit dem Heap-Speicher limitiert ist. Je mehr Anfragen gleichzeitig ankommen, desto wahrscheinlicher ist es, dass es hierbei zu Speicherproblemen kommt.

6.1.2.2 Validierung mittels Bean-Validierung

Ein anderer Ansatz der Validierung ist die Bean-Validierung, welche mit dem Java Specification Request (JSR)-303[79] gekommen ist. Dabei wird die SOAP-Anfrage mittels Java Architecture for XML Binding (JAXB) in eine Javaklasse konvertiert. Nach der Konvertierung werden die einzelnen Annotation aus dem JSR-303 mittels Validator überprüft. Dieser Ansatz klingt zwar gut, ist aber nicht ganz sauber, da die Struktur der übergebenen Anfrage nicht validiert wird, sondern nur die darin enthaltenen Daten. Man kann hierbei nun ein Feld mehrfach schicken, oder auch Felder hinzufügen, welche gar nicht definiert sind.

6.1.2.3 Gewählte Validierungslösung

Da weder der eine noch der andere Ansatz zufrieden stellend sind, wurde in der Implementierung eine klassische Validierung der Felder gewählt. Damit ist gemeint, dass nach der Konvertierung in eine Javaklasse die Felder einzeln überprüft werden. Die Validierung der gelieferten Daten erfolgt ebenfalls innerhalb des Systems und wurde mittels der Schemavaliderung von JAXB gelöst.

6.2 Was wurde nicht erreicht

Leider war das EESSI-System zum Ende dieser Arbeit noch nicht so weit, dass die Datenspezifikationen und die Schnittstellen fertig waren. Es war zwar nicht Teil dieser Arbeit, diesen Teil zu implementieren, jedoch wäre es interessant gewesen diesen noch zu analysieren.
Des weiteren wurde die Anforderung ANF016[80] nicht so umgesetzt, sondern als Gesamtlösung, da der derzeitige geplante Einsatz dies nicht erforderlich macht und so

[79] Mehr Informationen dazu unter `http://beanvalidation.org/1.0/`
[80] Jeder einzelne Schritt der Stapelverarbeitung soll separat getriggert werden können

die Kontrolle des Ablaufs innerhalb des Systems bleibt und nur der Startzeitpunkt von außen vorgegeben wird.

7 Fazit und Ausblick

7.1 Fazit

Zu Beginn der Arbeit wird auf die EU-Verordnung Nr. 883/2004 [Eurb] und die daraus resultierende Durchführungsverordnung (EG) Nr. 987/2009 [Eura] verwiesen, welche den Datenaustausch zwischen den einzelnen Staaten der EU regeln und neu bestimmen. Daraus ergab sich die Aufgabe, ein System zu entwickeln, welches an das EU-System angebunden werden kann.
Innerhalb der Analysephase wurde schon klar, dass dieses Unterfangen in der Grundthematik nicht das komplexeste ist, jedoch der Hund im Detail liegt. Zusammenfassend kann man sagen, dass, die in dieser Arbeit erörterten Punkte, das implementierte System im groben beschreiben und die wichtigsten Passagen genauer erörtert wurden. Dies sind die Abläufe der einzelnen Verarbeitungsschritte, der Aufbau der Webserviceschnittstellen und der Grundaufbau der Stapelverarbeitung.
Es wurde wären der Arbeit ein großer Wert auf die mögliche Erweiterbarkeit des Systems gelegt, wodurch das System flexibel gegenüber neuen Anforderungen im Bezug auf Ausgabe und Transformation bleibt. Hierbei bin ich auf die Entwicklung eines flexiblen XSL-FO-Konstruktes stolz, wobei sich nur der Aufruf der grundlegenden Elemente immer wiederholt. Da alle Payload-Elemente in ihren separaten Namespace liegen, hätte man auch alles in eine Datei packen können, was jedoch den weiteren Ausbau etwas erschweren würde.

7.2 Ausblick

Da die gewählten Formen der Konfiguration im Bereich der Transformation eher statisch sind, d.h., dass die XSL-FO-Stylesheets im Auslieferungspaket integriert sind, ist hier ein möglicher Erweiterungspunkt, dass man diese Art der Konfiguration auslagert und diese über eine geeignete Form z.B. in einer Datenbank hinterlegt, damit man nicht immer eine Auslieferung der Software machen muss, falls sich eine Transformation ändert. Mittels des Transformationsservices könnte man die geänderte Transformationsroutine sofort testen und im Anschluss in das Produktionsenvironment übernehmen, was die Dauer des Auslieferungsprozesses verkürzen würde.
Ein weiter Punkt hier ist natürlich die Anbindung an das EESSI-System, was zum heutigen Zeitpunkt mit Ende 2018 vorgesehen ist.
Abschließend soll noch erwähnt werden, dass der gewählte Ansatz zum Publizieren des Webservices, vielleicht nochmals überdacht wird, da die Möglichkeiten des Publizierens innerhalb eines Webarchivs vielleicht effizienter sind und die anderen Konfigurationsmöglichkeiten eine andere Form der Flexibilität zu lassen.
Als weitere Ausbaustufe des NG ist dazu noch eine Administrationsoberfläche geplant

und ein Reportingtool, welches für verschiedenste Auswertung verwendet werden soll.

Anhang A: Datentypen

Dieser Abschnitt beschreibt die Datentypen in den einzelnen Teilen des System, damit die grundlegenden Felder der einzelnen Objekte besser referenziert werden können.

A.1 SOAP-Datentypen

A.1.1 NachrichtSendenAnfrage

Name	Datentype	Multiplizität	Beschreibung
Sender	A.1.4	1	Wer versendet die Nachricht
EmpfängerListe	Liste von A.1.4	1..n	Wer soll die Nachricht empfangen
Geschäftsfalltyp	A.1.30	1	Welcher Geschäftsfalltyp wird verwendet
GeschäftsfalltypVersion	Text	1	Welche Version wird verwendet
Dokumententyp	A.1.31	1	Welches Dokument wird versendet
DokumententypVersion	Text	1	Welche Version wird verwendet
Anhangsliste	Liste von A.1.3	0..n	Welche Anhänge sollen verwendet werden
Payload	Binärdaten	1	Die Nachricht selbst

A.1.2 NachrichtSendenAntwort

Name	Datentype	Multiplizität	Beschreibung
NachrichtID	UUID	1	Die UUID der Nachricht innerhalb des Systems

A.1.3 AnhangReferenz

Name	Datentype	Multiplizität	Beschreibung
AnhangId	UUID	1	Die UUID des Anhangs

A.1.4 Institution

Name	Datentype	Multiplizität	Beschreibung
InstitutionID	Text	1	Die eindeutige ID innerhalb eines Staates
InstitutionAkronym	Text	1	Der Kurzname der Institution
InstitutionName	Text	1	Der Name der Institution
StaatenCode	A.1.5	1	Der 2-stellige Staatencode
StaatenName	Text	0..1	Name des Staates

A.1.5 StaatenCode

Generell werden die Staaten innerhalb des EESSI-Systems mittels des 2-stelligen Staatencodes nach ISO-3166-1 adressiert. Es gibt dazu 2 Ausnahmen:

- Griechenland: EL anstatt GR
- Großbritannien: UK anstatt GB (könnte nach dem Ausstieg nun wegfallen)

A.1.6 AnhangHerunterladenAnfrage

Name	Datentype	Multiplizität	Beschreibung
AnhangId	UUID	1	Die UUID des Anhangs
AnhangLöschen	Boolean	0..1	Kennzeichen, ob der Anhang aus dem System gelöscht werden soll

A.1.7 AnhangHerunterladenAnwort

Name	Datentype	Multiplizität	Beschreibung
Payload	Binärdaten	1	Der Anhang selbst

A.1.8 AnhangRaufladenAnfrage

MIME_Typ	Text	1	Der Datentyp des Anhangs, falls auch einmal andere Datentypen verarbeitet werden sollen
Payload	Binärdaten	1	Der Anhang selbst

A.1.9 AnhangRaufladenAntwort

Name	Datentype	Multiplizität	Beschreibung
AnhangId	UUID	1	Die UUID des Anhangs

A.1.10 ServiceException

Name	Datentype	Multiplizität	Beschreibung
ErrorCode	Text	1	Interner Code des Fehlers
ErrorNachricht	Text	1	Fehlerbeschreibung

A.1.11 NachrichtHerunterladenAnfrage

Name	Datentype	Multiplizität	Beschreibung
NachrichtID	UUID	1	Die UUID der Nachricht innerhalb des Systems
NachrichtLöschen	Boolean	0..1	Kennzeichen, ob die Nachricht aus dem System gelöscht werden soll

A.1.12 NachrichtHerunterladenAntwort

Name	Datentype	Multiplizität	Beschreibung
Payload	Binärdaten	1	Die Nachricht selbst

A.1.13 NachrichtenListeNachIDAnfrage

Name	Datentype	Multiplizität	Beschreibung
Institution	A.1.4	1	Die Institution, welche die Nachrichten empfangen hat
vonID	ganze Zahle	0..1	Ab dieser ID soll gesucht werden soll
bisID	ganze Zahle	0..1	Bis zu dieser ID soll gesucht werden
GeschäftsfalltypListe	Liste von A.1.30	1..n	Diese Geschäftsfalltypen sollen gesucht werden
Gelöscht	Boolean	0..1	Es sollen auch bereits gelöschte Nachrichten angezeigt werden

A.1.14 NachrichtenListeNachDatumAnfrage

Name	Datentype	Multiplizität	Beschreibung
Institution	A.1.4	1	Die Institution, welche die Nachrichten empfangen hat
vonDatum	Datum	0..1	Ab diesem Datum soll gesucht werden soll
bisDatum	Datum	0..1	Bis zu diesem Datum soll gesucht werden
GeschäftsfalltypListe	Liste von A.1.30	1..n	Diese Geschäftsfalltypen sollen gesucht werden
Gelöscht	Boolean	0..1	Es sollen auch bereits gelöschte Nachrichten angezeigt werden

A.1.15 NachrichtenListeAntwort

Name	Datentype	Multiplizität	Beschreibung
NachrichtenListe	Liste von A.1.16	0..1	Die Liste der Nachrichten nach den Suchkriterien

A.1.16 NachrichtMetadata

Name	Datentype	Multiplizität	Beschreibung
NachrichtNummer	Ganze Zahl	1	Die numerische ID der Nachricht, für den besseren Suchverhalt
NachrichtID	UUID	1	Die UUID der Nachricht innerhalb des Systems
Sender	A.1.4	1	Der Absender der Nachricht
Empfänger	A.1.4	1	Der Empfänger der Nachricht
Empfangsdatum	Datum	1	Datum, wann die Nachricht am NG eingegangen ist
Geschäftsfalltyp	A.1.30	1	Um welchen Geschäftsfalltyp handelt es sich
GeschäftsfalltypVersion	Text	1	Die Version des Geschäftsfalltyps
Dokumententyp	A.1.31	1	Um welches Dokument handelt es sich
DokumententypVersion	Text	1	Die Version des Dokuments
Gelöscht	Boolean	1	Zeigt an, ob die Nachricht bereits gelöscht wurde
Anhngliste	Liste von A.1.3	0..n	Eine mögliche Liste an Anhängen

A.1.17 InstitutionsAbfrageAnfrage

Name	Datentype	Multiplizität	Beschreibung
StaatenCode	A.1.5	1	Der 2-stellige Staatencode
InstitutionID	Text	0..1	Die eindeutige ID innerhalb eines Staates
InstitutionName	Text	0..1	Der Name der Institution oder ein Teil davon

A.1.18 ZuständigeInstitutionsAbfrageAnfrage

Name	Datentype	Multiplizität	Beschreibung
StaatenCode	A.1.5	1	Der 2-stellige Staatencode
Geschäftsfalltyp	A.1.30	1	Um welchen Geschäftsfalltyp handelt es sich
GeschäftsfalltypVersion	Text	1	Die Version des Geschäftsfalltyps

A.1.19 InstitutionsAbfrageAntwort

Name	Datentype	Multiplizität	Beschreibung
Institutionsliste	Liste von A.1.4	0..n	Die Liste der zutreffenden Institutionen

A.1.20 VerbindungsstelleAbfrageAnfrage

Name	Datentype	Multiplizität	Beschreibung
StaatenCode	A.1.5	1	Der 2-stellige Staatencode
Geschäftsfalltyp	A.1.30	1	Um welchen Geschäftsfalltyp handelt es sich
GeschäftsfalltypVersion	Text	1	Die Version des Geschäftsfalltyps

A.1.21 VerbindungsstelleAbfrageAntwort

Name	Datentype	Multiplizität	Beschreibung
Verbindungsstelle	A.1.4	0..1	Die entsprechende Verbindungsstelle

A.1.22 GeschäftsfalltypSucheMittelsGeschäftsfalltypAnfrage

Name	Datentype	Multiplizität	Beschreibung
Geschäftsfalltyp	A.1.30	1	Um welchen Geschäftsfalltyp handelt es sich
GeschäftsfalltypVersion	Text	1	Die Version des Geschäftsfalltyps

A.1.23 GeschäftsfalltypSucheMittelsDokumententypAnfrage

Name	Datentype	Multiplizität	Beschreibung
Dokumententyp	A.1.31	1	Um welchen Geschäftsfalltyp handelt es sich
DokumententypVersion	Text	1	Die Version des Geschäftsfalltyps

A.1.24 GeschäftsfalltypSucheAntwort

Name	Datentype	Multiplizität	Beschreibung
Geschäftsfalltyp	A.1.30	1	Um welchen Geschäftsfalltyp handelt es sich
GeschäftsfalltypVersion	Text	1	Die Version des Geschäftsfalltyps
Liste von Dokumententypen	Liste von A.1.25	1..n	Die Metadaten der einzelnen Dokmententypen

A.1.25 Dokumententyp

Name	Datentype	Multiplizität	Beschreibung
Dokumententyp	A.1.31	1	Um welchen Geschäftsfalltyp handelt es sich
DokumententypVersion	Text	1	Die Version des Geschäftsfalltyps
InitialDokument	Boolean	1	Gibt an, ob es sich um das erste zu sendende Dokument handelt
DokumentGesendetVonStarter	Boolean	1	Gibt an, ob es sich um das Dokument vom Initiator gesendet werden darf
DokumentGesendetvonGegenpartei	Boolean	1	Gibt an, ob es sich um das Dokument vom Partnerinstitut gesendet werden darf

A.1.26 NachrichtTransformierenAnfrage

Name	Datentype	Multiplizität	Beschreibung
Sender	A.1.4	1	Wer versendet die Nachricht
Geschäftsfalltyp	A.1.30	1	Welcher Geschäftsfalltyp wird verwendet
GeschäftsfalltypVersion	Text	1	Welche Version wird verwendet
Dokumententyp	A.1.31	1	Welches Dokument wird versendet
DokumententypVersion	Text	1	Welche Version wird verwendet
Anhangsliste	Liste von A.1.3	0..n	Welche Anhänge sollen verwendet werden
Payload	Binärdaten	1	Die Nachricht selbst

A.1.27 NachrichtTransformierenAntwort

Name	Datentype	Multiplizität	Beschreibung
NachrichtID	UUID	1	Die UUID der Nachricht innerhalb des Systems

A.1.28 TransformierteNachrichtHerunterladenAnfrage

Name	Datentype	Multiplizität	Beschreibung
Sender	A.1.4	1	Wer versendet die Nachricht
Geschäftsfalltyp	A.1.30	1	Welcher Geschäftsfalltyp wird verwendet
GeschäftsfalltypVersion	Text	1	Welche Version wird verwendet
Dokumententyp	A.1.31	1	Welches Dokument wird versendet
DokumententypVersion	Text	1	Welche Version wird verwendet
NachrichtID	UUID	1	Die UUID der Nachricht innerhalb des Systems

A.1.29 TransformierteNachrichtHerunterladenAntwort

Name	Datentype	Multiplizität	Beschreibung
Payload	Binärdaten	1	Die transformierte Nachricht selbst

A.1.30 Geschäftsfalltyp

Name	Datentype	Multiplizität	Beschreibung
Geschäftsfalltyp	Text	1	Die ID eines Geschäftfalls

A.1.31 Dokumententyp

Name	Datentype	Multiplizität	Beschreibung
Dokumententyp	Text	1	Die ID eines Dokuments

Anhang B: Quelltexte

B.1 Verwendete Datentypen im WebService

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  targetNamespace="http://gateway.nat/messages/typen/"
  xmlns:ng="http://gateway.nat/messages/typen/"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
    >
  <xs:annotation>
    <xs:documentation>Diese Schemadatei beinhaltet alle Datentypen</xs:
        documentation>
  </xs:annotation>
  <xs:complexType name="NachrichtSendenAnfrageTyp">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="sender" type="ng:
          InstitutionTyp"/>
      <xs:element minOccurs="1" maxOccurs="1" name="empfaengerliste" type="ng:
          EmpfaengerListTyp"/>
      <xs:element minOccurs="0" maxOccurs="1" name="empfaengerlistecc" type="ng:
          EmpfaengerListTyp"/>
      <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
          string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
          type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
          string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
          xs:string"/>
      <xs:element minOccurs="0" maxOccurs="1" name="anhangliste" type="ng:
          AnhanglistTyp"/>
      <xs:element minOccurs="1" maxOccurs="1" name="payload" type="xs:
          base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AnhanglistTyp">
    <xs:choice>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="anhang" type="ng:
          Anhangtyp"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="Anhangtyp">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="mimeTyp" type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="anhangid" type="ng:UUIDType"/
```

```
        >
35    </xs:sequence>
36  </xs:complexType>
37  <xs:complexType name="InstitutionTyp">
38    <xs:sequence>
39      <xs:element minOccurs="1" maxOccurs="1" name="institutionID" type="xs:
            string"/>
40      <xs:element minOccurs="0" maxOccurs="1" name="institutionAkronym" type="xs:
            string"/>
41      <xs:element minOccurs="0" maxOccurs="1" name="InstitutionName" type="xs:
            string"/>
42      <xs:element minOccurs="1" maxOccurs="1" name="staatenCode" type="xs:string"
            />
43      <xs:element minOccurs="0" maxOccurs="1" name="staatenName" type="xs:string"
            />
44    </xs:sequence>
45  </xs:complexType>
46  <xs:complexType name="EmpfaengerListTyp">
47    <xs:sequence>
48      <xs:element minOccurs="1" maxOccurs="unbounded" name="institute" type="ng:
            InstitutionTyp"/>
49    </xs:sequence>
50  </xs:complexType>
51
52  <xs:complexType name="NachrichtSendenAntwortTyp">
53    <xs:sequence>
54      <xs:element minOccurs="1" maxOccurs="1" name="nachrichtId" type="ng:
            UUIDType"/>
55    </xs:sequence>
56  </xs:complexType>
57
58  <xs:simpleType name="UUIDType">
59    <xs:restriction base="xs:string">
60      <xs:length value="36" fixed="true"/>
61      <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F
            ]{4}-[0-9a-fA-F]{12}"/>
62    </xs:restriction>
63  </xs:simpleType>
64
65  <xs:complexType name="AnhangHerunterladenAnfrageTyp">
66    <xs:sequence>
67      <xs:element minOccurs="1" maxOccurs="1" name="anhangId" type="ng:UUIDType"/
            >
68      <xs:element minOccurs="0" maxOccurs="1" name="anhangLoeschen" type="xs:
            boolean"/>
69    </xs:sequence>
70  </xs:complexType>
71
72  <xs:complexType name="AnhangHerunterladenAntwortTyp">
73    <xs:sequence>
74      <xs:element minOccurs="1" maxOccurs="1" name="payload" type="xs:
            base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
```

```
75      </xs:sequence>
76    </xs:complexType>
77
78    <xs:complexType name="AnhangRaufladenAntwortTyp">
79      <xs:sequence>
80        <xs:element name="anhangId" type="ng:UUIDType"/>
81      </xs:sequence>
82    </xs:complexType>
83
84    <xs:complexType name="AnhangRaufladenAnfrageTyp">
85      <xs:sequence>
86        <xs:element minOccurs="1" maxOccurs="1" name="mimeType" type="xs:string"/>
87        <xs:element minOccurs="1" maxOccurs="1" name="payload" type="xs:
              base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
88      </xs:sequence>
89    </xs:complexType>
90
91    <xs:complexType name="NachrichtHerunterladenAnfrageTyp">
92      <xs:sequence>
93        <xs:element minOccurs="1" maxOccurs="1" name="nachrichtenId" type="xs:long"
              />
94        <xs:element minOccurs="0" maxOccurs="1" name="keep" type="xs:boolean"/>
95      </xs:sequence>
96    </xs:complexType>
97    <xs:complexType name="NachrichtHerunterladenAntwortTyp">
98      <xs:sequence>
99        <xs:element minOccurs="1" name="payload" type="xs:base64Binary" xmime:
              contentType="application/octet-stream" xmime:expectedContentTypes="
              application/octet-stream"/>
100     </xs:sequence>
101   </xs:complexType>
102
103   <xs:complexType name="NachrichtenListeNachIDAnfrageTyp">
104     <xs:sequence>
105       <xs:element minOccurs="1" maxOccurs="1" name="institution" type="ng:
              InstitutionTyp"/>
106       <xs:element minOccurs="0" maxOccurs="1" name="vonId" type="xs:long"/>
107       <xs:element minOccurs="0" maxOccurs="1" name="bisId" type="xs:long"/>
108       <xs:element minOccurs="0" maxOccurs="1" name="geschaeftsfalltypListe" type=
              "ng:GeschaeftsfalltypListTyp"/>
109       <xs:element minOccurs="0" maxOccurs="1" name="geloescht" type="xs:boolean"/
              >
110     </xs:sequence>
111   </xs:complexType>
112   <xs:complexType name="NachrichtenListeNachDatumAnfrageTyp">
113     <xs:sequence>
114       <xs:element minOccurs="1" maxOccurs="1" name="institution" type="ng:
              InstitutionTyp"/>
115       <xs:element minOccurs="1" maxOccurs="1" name="vonDatum" type="xs:dateTime"/
              >
116       <xs:element minOccurs="0" maxOccurs="1" name="bisDatum" type="xs:dateTime"/
              >
```

```
117       <xs:element minOccurs="0" maxOccurs="1" name="GeschaeftsfalltypListe" type=
              "ng:GeschaeftsfalltypListTyp"/>
118       <xs:element minOccurs="0" maxOccurs="1" name="geloescht" type="xs:boolean"/
              >
119     </xs:sequence>
120   </xs:complexType>
121
122   <xs:complexType name="GeschaeftsfalltypListTyp">
123     <xs:sequence>
124       <xs:element minOccurs="1" maxOccurs="unbounded" name="Geschaeftsfalltyp"
              type="xs:string"/>
125     </xs:sequence>
126   </xs:complexType>
127
128   <xs:complexType name="NachrichtenListeAntwortTyp">
129     <xs:sequence>
130       <xs:element maxOccurs="unbounded" minOccurs="0" name="messageMetaData" type
              ="ng:NachrichtMetaDataTyp"/>
131     </xs:sequence>
132   </xs:complexType>
133   <xs:complexType name="NachrichtMetaDataTyp">
134     <xs:sequence>
135       <xs:element minOccurs="1" maxOccurs="1" name="nachrichtNummer" type="xs:
              long"/>
136       <xs:element minOccurs="1" maxOccurs="1" name="nachrichtId" type="ng:
              UUIDType"/>
137       <xs:element minOccurs="1" maxOccurs="1" name="sender" type="ng:
              InstitutionTyp"/>
138       <xs:element minOccurs="1" maxOccurs="1" name="empfaenger" type="ng:
              InstitutionTyp"/>
139       <xs:element minOccurs="1" maxOccurs="1" name="empfangsdatum" type="xs:
              dateTime"/>
140       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
              string"/>
141       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
              type="xs:string"/>
142       <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
              string"/>
143       <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
              xs:string"/>
144       <xs:element minOccurs="1" maxOccurs="1" name="geloescht" type="xs:boolean"/
              >
145       <xs:element minOccurs="0" maxOccurs="1" name="anhangListe" type="ng:
              AnhanglistTyp"/>
146     </xs:sequence>
147   </xs:complexType>
148
149   <xs:complexType name="InstitutionsAbfrageAntwortTyp">
150     <xs:sequence>
151       <xs:element maxOccurs="unbounded" minOccurs="0" name="institute" type="ng:
              InstitutionTyp"/>
152     </xs:sequence>
```

```
153   </xs:complexType>
154
155   <xs:complexType name="InstitutionsAbfrageAnfrageTyp">
156     <xs:sequence>
157       <xs:element minOccurs="1" maxOccurs="1" name="staatenCode" type="xs:string"
          />
158       <xs:element minOccurs="0" maxOccurs="1" name="institutionId" type="xs:
          string"/>
159       <xs:element minOccurs="0" maxOccurs="1" name="institutionName" type="xs:
          string"/>
160     </xs:sequence>
161   </xs:complexType>
162
163   <xs:complexType name="VerbindungsstelleAbfrageAntwortTyp">
164     <xs:sequence>
165       <xs:element minOccurs="0" maxOccurs="1" name="verbindungsstelle" type="ng:
          InstitutionTyp"/>
166     </xs:sequence>
167   </xs:complexType>
168
169   <xs:complexType name="VerbindungsstelleAbfrageAnfrageTyp">
170     <xs:sequence>
171       <xs:element minOccurs="1" maxOccurs="1" name="staatenCode" type="xs:string"
          />
172       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
          string"/>
173       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
          type="xs:string"/>
174     </xs:sequence>
175   </xs:complexType>
176
177   <xs:complexType name="GeschaeftsfalltypSucheAntwortTyp">
178     <xs:sequence>
179       <xs:element minOccurs="0" maxOccurs="1" name="geschaeftsfallMetaDaten" type
          ="ng:GeschaeftsfallMetaDatenTyp"/>
180     </xs:sequence>
181   </xs:complexType>
182
183   <xs:complexType name="GeschaeftsfallMetaDatenTyp">
184     <xs:sequence>
185       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
          string"/>
186       <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
          type="xs:string"/>
187       <xs:element minOccurs="1" maxOccurs="unbounded" name="dokumententyp" type="
          ng:DokumententypMetaDatenTyp"/>
188     </xs:sequence>
189   </xs:complexType>
190
191   <xs:complexType name="DokumententypMetaDatenTyp">
192     <xs:sequence>
193       <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
```

```
            string"/>
194         <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
                xs:string"/>
195         <xs:element minOccurs="1" maxOccurs="1" name="initialDokument" type="xs:
                boolean"/>
196         <xs:element minOccurs="1" maxOccurs="1" name="dokumentGesendetVonStarter"
                type="xs:boolean"/>
197         <xs:element minOccurs="1" maxOccurs="1" name="
                dokumentGesendetVonGegenpartei" type="xs:boolean"/>
198       </xs:sequence>
199     </xs:complexType>
200   <!-- Hier mach weiter -->
201     <xs:complexType name="GeschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrageTyp
            ">
202       <xs:sequence>
203         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
                string"/>
204         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
                type="xs:string"/>
205       </xs:sequence>
206     </xs:complexType>
207
208     <xs:complexType name="GeschaeftsfalltypSucheMittelsDokumententypAnfrageTyp">
209       <xs:sequence>
210         <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
                string"/>
211         <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
                xs:string"/>
212       </xs:sequence>
213     </xs:complexType>
214
215     <!-- Neues Feld flowVersion -->
216     <xs:complexType name="ZustaendigeInstitutionsAbfrageAnfrageTyp">
217       <xs:sequence>
218         <xs:element minOccurs="1" maxOccurs="1" name="staatenCode" type="xs:string"
                />
219         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
                string"/>
220         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
                type="xs:string"/>
221       </xs:sequence>
222     </xs:complexType>
223
224     <xs:complexType name="NachrichtTransformierenAnfrageTyp">
225       <xs:sequence>
226         <xs:element minOccurs="1" maxOccurs="1" name="sender" type="ng:
                InstitutionTyp"/>
227         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
                string"/>
228         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
                type="xs:string"/>
229         <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
```

```
            string"/>
230         <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
                xs:string"/>
231         <xs:element minOccurs="0" maxOccurs="1" name="anhangListe" type="ng:
                AnhanglistTyp"/>
232         <xs:element minOccurs="1" maxOccurs="1" name="payload" type="xs:
                base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
233       </xs:sequence>
234     </xs:complexType>
235
236     <xs:complexType name="NachrichtTransformierenAntwortTyp">
237       <xs:sequence>
238         <xs:element minOccurs="1" maxOccurs="1" name="uuid" type="ng:UUIDType"/>
239       </xs:sequence>
240     </xs:complexType>
241
242     <xs:complexType name="TransformierteNachrichtHerunterladenAnfrageTyp">
243       <xs:sequence>
244         <xs:element minOccurs="1" maxOccurs="1" name="sender" type="ng:
                InstitutionTyp"/>
245         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltyp" type="xs:
                string"/>
246         <xs:element minOccurs="1" maxOccurs="1" name="geschaeftsfalltypVersion"
                type="xs:string"/>
247         <xs:element minOccurs="1" maxOccurs="1" name="dokumententyp" type="xs:
                string"/>
248         <xs:element minOccurs="1" maxOccurs="1" name="dokumententypVersion" type="
                xs:string"/>
249         <xs:element minOccurs="1" maxOccurs="1" name="nachrichtId" type="ng:
                UUIDType"/>
250       </xs:sequence>
251     </xs:complexType>
252
253     <xs:complexType name="TransformierteNachrichtHerunterladenAntwortTyp">
254       <xs:sequence>
255         <xs:element minOccurs="1" maxOccurs="1" name="pdfData" type="xs:
                base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
256       </xs:sequence>
257     </xs:complexType>
258
259   <xs:complexType name="ServiceExceptionType">
260     <xs:sequence>
261       <xs:element name="ErrorCode" type="xs:string"/>
262       <xs:element name="ErrorMessage" type="xs:string"/>
263     </xs:sequence>
264   </xs:complexType>
265   </xs:schema>
```

Quelltext B.1: NationalesGateway.xsd

B.2 WSDL - Informationsservice

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions targetNamespace="http://gateway.nat/messages/"
                  xmlns:typen="http://gateway.nat/messages/typen/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://gateway.nat/messages/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
  <xs:annotation>

    <xs:documentation>
      Hier sind alle Informationsservices gelistet.
    </xs:documentation>
  </xs:annotation>
  <wsdl:types>
    <xs:schema targetNamespace="http://gateway.nat/messages/">
      <xs:import schemaLocation="NationalesGateway.xsd" namespace="http://gateway
          .nat/messages/typen/"/>
      <xs:element type="typen:InstitutionsAbfrageAntwortTyp" name="
          institutionsAbfrageAntwort"/>
      <xs:element type="typen:InstitutionsAbfrageAnfrageTyp" name="
          institutionsAbfrageAnfrage"/>
      <xs:element type="typen:VerbindungsstelleAbfrageAntwortTyp" name="
          verbindungsstelleAbfrageAntwort"/>
      <xs:element type="typen:VerbindungsstelleAbfrageAnfrageTyp" name="
          verbindungsstelleAbfrageAnfrage"/>
      <xs:element type="typen:ZustaendigeInstitutionsAbfrageAnfrageTyp" name="
          zustaendigeInstitutionsAbfrageAnfrage"/>
      <xs:element type="typen:GeschaeftsfalltypSucheAntwortTyp" name="
          geschaeftsfalltypSucheAntwort"/>
      <xs:element type="typen:
          GeschaeftsfalltypSucheMittelsDokumententypAnfrageTyp" name="
          geschaeftsfalltypSucheMittelsDokumententypAnfrage"/>
      <xs:element type="typen:
          GeschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrageTyp" name="
          geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage"/>
      <xs:element type="typen:ServiceExceptionType" name="serviceException"/>
    </xs:schema>
  </wsdl:types>

  <!-- Reading Institutions -->
  <wsdl:message name="institutionsAbfrageAntwort">
    <wsdl:part element="tns:institutionsAbfrageAntwort" name="
        institutionsAbfrageAntwort"/>
  </wsdl:message>
  <wsdl:message name="institutionsAbfrageAnfrage">
    <wsdl:part element="tns:institutionsAbfrageAnfrage" name="
        institutionsAbfrageAnfrage"/>
  </wsdl:message>
  <wsdl:message name="verbindungsstelleAbfrageAntwort">
    <wsdl:part element="tns:verbindungsstelleAbfrageAntwort" name="
        verbindungsstelleAbfrageAntwort"/>
```

```
39  </wsdl:message>
40  <wsdl:message name="verbindungsstelleAbfrageAnfrage">
41    <wsdl:part element="tns:verbindungsstelleAbfrageAnfrage" name="
        verbindungsstelleAbfrageAnfrage"/>
42  </wsdl:message>
43
44  <!-- Flow Attributions -->
45  <wsdl:message name="zustaendigeInstitutionsAbfrageAnfrage">
46    <wsdl:part element="tns:zustaendigeInstitutionsAbfrageAnfrage" name="
        zustaendigeInstitutionsAbfrageAnfrage">
47    </wsdl:part>
48  </wsdl:message>
49
50  <!-- Flow Context -->
51  <wsdl:message name="geschaeftsfalltypSucheAntwort">
52    <wsdl:part element="tns:geschaeftsfalltypSucheAntwort" name="
        geschaeftsfalltypSucheAntwort"/>
53  </wsdl:message>
54
55  <wsdl:message name="geschaeftsfalltypSucheMittelsDokumententypAnfrage">
56    <wsdl:part element="tns:geschaeftsfalltypSucheMittelsDokumententypAnfrage"
        name="geschaeftsfalltypSucheMittelsDokumententypAnfrage"/>
57  </wsdl:message>
58  <wsdl:message name="geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage">
59    <wsdl:part element="tns:geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage
        " name="geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage"/>
60  </wsdl:message>
61
62  <wsdl:message name="ServiceException">
63    <wsdl:part element="tns:serviceException" name="ServiceException"/>
64  </wsdl:message>
65
66  <wsdl:portType name="InformationServicePort">
67    <wsdl:operation name="institutionsAbfrage">
68      <wsdl:input message="tns:institutionsAbfrageAnfrage" name="
          institutionsAbfrageAnfrage"/>
69      <wsdl:output message="tns:institutionsAbfrageAntwort" name="
          institutionsAbfrageAntwort"/>
70      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
71    </wsdl:operation>
72
73    <wsdl:operation name="verbindungsstelleAbfrage">
74      <wsdl:input message="tns:verbindungsstelleAbfrageAnfrage" name="
          verbindungsstelleAbfrageAnfrage"/>
75      <wsdl:output message="tns:verbindungsstelleAbfrageAntwort" name="
          verbindungsstelleAbfrageAntwort"/>
76      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
77    </wsdl:operation>
78
79    <wsdl:operation name="geschaeftsfalltypSucheMittelsGeschaeftsfalltyp">
80      <wsdl:input message="tns:
          geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage" name="
```

```
        geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage"/>
81      <wsdl:output message="tns:geschaeftsfalltypSucheAntwort" name="
            geschaeftsfalltypSucheAntwort"/>
82      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
83    </wsdl:operation>
84    <wsdl:operation name="geschaeftsfalltypSucheMittelsDokumententyp">
85      <wsdl:input message="tns:geschaeftsfalltypSucheMittelsDokumententypAnfrage"
             name="geschaeftsfalltypSucheMittelsDokumententypAnfrage"/>
86      <wsdl:output message="tns:geschaeftsfalltypSucheAntwort" name="
            geschaeftsfalltypSucheAntwort"/>
87      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
88    </wsdl:operation>
89
90    <wsdl:operation name="zustaendigeInstitutionsAbfrage">
91      <wsdl:input message="tns:zustaendigeInstitutionsAbfrageAnfrage" name="
            zustaendigeInstitutionsAbfrageAnfrage"/>
92      <wsdl:output message="tns:institutionsAbfrageAntwort" name="
            institutionsAbfrageAntwort"/>
93      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
94    </wsdl:operation>
95
96  </wsdl:portType>
97
98  <wsdl:binding name="InformationServiceSoapBinding" type="tns:
        InformationServicePort">
99    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
          http"/>
100
101   <wsdl:operation name="institutionsAbfrage">
102     <soap:operation soapAction="" style="document"/>
103     <wsdl:input name="institutionsAbfrageAnfrage">
104       <soap:body use="literal"/>
105     </wsdl:input>
106     <wsdl:output name="institutionsAbfrageAntwort">
107       <soap:body use="literal"/>
108     </wsdl:output>
109     <wsdl:fault name="ServiceException">
110       <soap:fault name="ServiceException" use="literal"/>
111     </wsdl:fault>
112   </wsdl:operation>
113
114   <wsdl:operation name="verbindungsstelleAbfrage">
115     <soap:operation soapAction="" style="document"/>
116     <wsdl:input name="verbindungsstelleAbfrageAnfrage">
117       <soap:body use="literal"/>
118     </wsdl:input>
119     <wsdl:output name="verbindungsstelleAbfrageAntwort">
120       <soap:body use="literal"/>
121     </wsdl:output>
122     <wsdl:fault name="ServiceException">
123       <soap:fault name="ServiceException" use="literal"/>
124     </wsdl:fault>
```

```
125     </wsdl:operation>
126
127     <wsdl:operation name="geschaeftsfalltypSucheMittelsGeschaeftsfalltyp">
128       <soap:operation soapAction="" style="document"/>
129       <wsdl:input name="geschaeftsfalltypSucheMittelsGeschaeftsfalltypAnfrage">
130         <soap:body use="literal"/>
131       </wsdl:input>
132       <wsdl:output name="geschaeftsfalltypSucheAntwort">
133         <soap:body use="literal"/>
134       </wsdl:output>
135       <wsdl:fault name="ServiceException">
136         <soap:fault name="ServiceException" use="literal"/>
137       </wsdl:fault>
138     </wsdl:operation>
139     <wsdl:operation name="geschaeftsfalltypSucheMittelsDokumententyp">
140       <soap:operation soapAction="" style="document"/>
141       <wsdl:input name="geschaeftsfalltypSucheMittelsDokumententypAnfrage">
142         <soap:body use="literal"/>
143       </wsdl:input>
144       <wsdl:output name="geschaeftsfalltypSucheAntwort">
145         <soap:body use="literal"/>
146       </wsdl:output>
147       <wsdl:fault name="ServiceException">
148         <soap:fault name="ServiceException" use="literal"/>
149       </wsdl:fault>
150     </wsdl:operation>
151
152     <wsdl:operation name="zustaendigeInstitutionsAbfrage">
153       <soap:operation soapAction="" style="document"/>
154       <wsdl:input name="zustaendigeInstitutionsAbfrageAnfrage">
155         <soap:body use="literal"/>
156       </wsdl:input>
157       <wsdl:output name="institutionsAbfrageAntwort">
158         <soap:body use="literal"/>
159       </wsdl:output>
160       <wsdl:fault name="ServiceException">
161         <soap:fault name="ServiceException" use="literal"/>
162       </wsdl:fault>
163     </wsdl:operation>
164   </wsdl:binding>
165   <wsdl:service name="InformationService">
166     <wsdl:port binding="tns:InformationServiceSoapBinding" name="
          InformationServicePort">
167       <soap:address location="TO BE DEFINED"/>
168     </wsdl:port>
169   </wsdl:service>
170 </wsdl:definitions>
```

Quelltext B.2: InformationService.wsdl

B.3 WSDL - Transformationsservice

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions targetNamespace="http://gateway.nat/messages/"
                  xmlns:typen="http://gateway.nat/messages/typen/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://gateway.nat/messages/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:jaxws="http://java.sun.com/xml/ns/jaxws"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xs:schema targetNamespace="http://gateway.nat/messages/">
      <xs:import schemaLocation="NationalesGateway.xsd" namespace="http://gateway
          .nat/messages/typen/" />
      <xs:element type="typen:TransformierteNachrichtHerunterladenAnfrageTyp"
          name="transformierteNachrichtHerunterladenAnfrage"/>
      <xs:element type="typen:TransformierteNachrichtHerunterladenAntwortTyp"
          name="transformierteNachrichtHerunterladenAntwort"/>
      <xs:element type="typen:NachrichtTransformierenAnfrageTyp" name="
          nachrichtTransformierenAnfrage"/>
      <xs:element type="typen:NachrichtTransformierenAntwortTyp" name="
          nachrichtTransformierenAntwort"/>
      <xs:element type="typen:ServiceExceptionType" name="serviceException"/>

    </xs:schema>
  </wsdl:types>

  <wsdl:message name="nachrichtTransformierenAntwort">
    <wsdl:part element="tns:nachrichtTransformierenAntwort" name="
        nachrichtTransformierenAntwort"/>
  </wsdl:message>
  <wsdl:message name="nachrichtTransformierenAnfrage">
    <wsdl:part element="tns:nachrichtTransformierenAnfrage" name="
        nachrichtTransformierenAnfrage"/>
  </wsdl:message>
  <wsdl:message name="transformierteNachrichtHerunterladenAntwort">
    <wsdl:part element="tns:transformierteNachrichtHerunterladenAntwort" name="
        transformierteNachrichtHerunterladenAntwort"/>
  </wsdl:message>
  <wsdl:message name="transformierteNachrichtHerunterladenAnfrage">
    <wsdl:part element="tns:transformierteNachrichtHerunterladenAnfrage" name="
        transformierteNachrichtHerunterladenAnfrage"/>
  </wsdl:message>

  <wsdl:message name="ServiceException">
    <wsdl:part element="tns:serviceException" name="serviceException"/>
  </wsdl:message>

  <wsdl:portType name="TransformServicesPort">
    <wsdl:operation name="nachrichtTransformieren">
      <wsdl:input message="tns:nachrichtTransformierenAnfrage" name="
          nachrichtTransformierenAnfrage"/>
      <wsdl:output message="tns:nachrichtTransformierenAntwort" name="
          nachrichtTransformierenAntwort"/>
```

```
42      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
43    </wsdl:operation>
44    <wsdl:operation name="transformierteNachrichtHerunterladen">
45      <wsdl:input message="tns:transformierteNachrichtHerunterladenAnfrage" name=
            "transformierteNachrichtHerunterladenAnfrage"/>
46      <wsdl:output message="tns:transformierteNachrichtHerunterladenAntwort" name
            ="transformierteNachrichtHerunterladenAntwort"/>
47      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
48    </wsdl:operation>
49  </wsdl:portType>
50
51  <wsdl:binding name="TransformServiceSoapBinding" type="tns:
        TransformServicesPort">
52    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
          http"/>
53    <wsdl:operation name="nachrichtTransformieren">
54      <soap:operation soapAction="" style="document"/>
55      <wsdl:input name="nachrichtTransformierenAnfrage">
56        <soap:body use="literal"/>
57      </wsdl:input>
58      <wsdl:output name="nachrichtTransformierenAntwort">
59        <soap:body use="literal"/>
60      </wsdl:output>
61      <wsdl:fault name="ServiceException">
62        <soap:fault name="ServiceException" use="literal"/>
63      </wsdl:fault>
64    </wsdl:operation>
65    <wsdl:operation name="transformierteNachrichtHerunterladen">
66      <soap:operation soapAction="" style="document"/>
67      <wsdl:input name="transformierteNachrichtHerunterladenAnfrage">
68        <soap:body use="literal"/>
69      </wsdl:input>
70      <wsdl:output name="transformierteNachrichtHerunterladenAntwort">
71        <soap:body use="literal"/>
72      </wsdl:output>
73      <wsdl:fault name="ServiceException">
74        <soap:fault name="ServiceException" use="literal"/>
75      </wsdl:fault>
76    </wsdl:operation>
77  </wsdl:binding>
78
79  <wsdl:service name="TransformService">
80    <wsdl:port binding="tns:TransformServiceSoapBinding" name="
          TransformServicePort">
81      <soap:address location="TO BE DEFINED"/>
82    </wsdl:port>
83  </wsdl:service>
84 </wsdl:definitions>
```

Quelltext B.3: TransformService.wsdl

B.4 WSDL - SendeService

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions targetNamespace="http://gateway.nat/messages/"
  xmlns:types="http://gateway.nat/messages/typen/"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                xmlns:tns="http://gateway.nat/messages/"
                xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xs:schema targetNamespace="http://gateway.nat/messages/">
      <xs:import schemaLocation="NationalesGateway.xsd" namespace="http://gateway
          .nat/messages/typen/" />
      <xs:element type="types:AnhangRaufladenAnfrageTyp" name="
          anhangRaufladenAnfrage"/>
      <xs:element type="types:AnhangRaufladenAntwortTyp" name="
          anhangRaufladenAntwort"/>
      <xs:element type="types:NachrichtSendenAnfrageTyp" name="
          nachrichtSendenAnfrage"/>
      <xs:element type="types:NachrichtSendenAntwortTyp" name="
          nachrichtSendenAntwort"/>
      <xs:element type="types:ServiceExceptionType" name="serviceException"/>

    </xs:schema>
  </wsdl:types>

  <wsdl:message name="anhangRaufladenAnfrageMessage">
    <wsdl:part element="tns:anhangRaufladenAnfrage" name="anhangRaufladenAnfrage"
        />
  </wsdl:message>
  <wsdl:message name="anhangRaufladenAntwortMessage">
    <wsdl:part element="tns:anhangRaufladenAntwort" name="anhangRaufladenAntwort"
        />
  </wsdl:message>


  <wsdl:message name="nachrichtSendenAnfrage">
    <wsdl:part element="tns:nachrichtSendenAnfrage" name="nachrichtSendenAnfrage"
        />
  </wsdl:message>
  <wsdl:message name="nachrichtSendenAntwort">
    <wsdl:part element="tns:nachrichtSendenAntwort" name="nachrichtSendenAntwort"
        />
  </wsdl:message>
  <wsdl:message name="ServiceException">
    <wsdl:part element="tns:serviceException" name="serviceException"/>
  </wsdl:message>

  <wsdl:portType name="SendeServicesPort">
    <wsdl:operation name="nachrichtSenden">
      <wsdl:input message="tns:nachrichtSendenAnfrage" name="
          nachrichtSendenAnfrage"/>
```

```
41      <wsdl:output message="tns:nachrichtSendenAntwort" name="
            nachrichtSendenAntwort"/>
42      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
43    </wsdl:operation>
44
45    <wsdl:operation name="anhangRaufladen">
46      <wsdl:input message="tns:anhangRaufladenAnfrageMessage" name="
            anhangRaufladenAnfrage"/>
47      <wsdl:output message="tns:anhangRaufladenAntwortMessage" name="
            anhangRaufladenAntwort"/>
48      <wsdl:fault message="tns:ServiceException" name="ServiceException"/>
49    </wsdl:operation>
50
51  </wsdl:portType>
52
53  <wsdl:binding name="SendeServiceSoapBinding" type="tns:SendeServicesPort">
54    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
          http"/>
55    <wsdl:operation name="nachrichtSenden">
56      <soap:operation soapAction="" style="document"/>
57      <wsdl:input name="nachrichtSendenAnfrage">
58        <soap:body use="literal"/>
59      </wsdl:input>
60      <wsdl:output name="nachrichtSendenAntwort">
61        <soap:body use="literal"/>
62      </wsdl:output>
63      <wsdl:fault name="ServiceException">
64        <soap:fault name="ServiceException" use="literal"/>
65      </wsdl:fault>
66    </wsdl:operation>
67    <wsdl:operation name="anhangRaufladen">
68      <soap:operation soapAction="" style="document"/>
69      <wsdl:input name="anhangRaufladenAnfrage">
70        <soap:body use="literal"/>
71      </wsdl:input>
72      <wsdl:output name="anhangRaufladenAntwort">
73        <soap:body use="literal"/>
74      </wsdl:output>
75      <wsdl:fault name="ServiceException">
76        <soap:fault name="ServiceException" use="literal"/>
77      </wsdl:fault>
78    </wsdl:operation>
79  </wsdl:binding>
80
81  <wsdl:service name="SendeService">
82    <wsdl:port binding="tns:SendeServiceSoapBinding" name="SendeServicePort">
83      <soap:address location="TO BE DEFINED"/>
84    </wsdl:port>
85  </wsdl:service>
86 </wsdl:definitions>
```

Quelltext B.4: SendeService.wsdl

B.5 WSDL - EmpfangeService

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions targetNamespace="http://gateway.nat/messages/"

                  xmlns:typen="http://gateway.nat/messages/typen/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://gateway.nat/messages/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    >
  <wsdl:types>
    <xs:schema targetNamespace="http://gateway.nat/messages/">
      <xs:import schemaLocation="NationalesGateway.xsd" namespace="http://gateway
          .nat/messages/typen/" />
      <xs:element type="typen:NachrichtHerunterladenAnfrageTyp" name="
          nachrichtHerunterladenAnfrage" />
      <xs:element type="typen:NachrichtHerunterladenAntwortTyp" name="
          nachrichtHerunterladenAntwort" />
      <xs:element type="typen:AnhangHerunterladenAnfrageTyp" name="
          anhangHerunterladenAnfrage"/>
      <xs:element type="typen:AnhangHerunterladenAntwortTyp" name="
          anhangHerunterladenAntwort"/>
      <xs:element type="typen:NachrichtenListeAntwortTyp" name="
          nachrichtenListeAntwort" />
      <xs:element type="typen:NachrichtenListeNachDatumAnfrageTyp" name="
          nachrichtenListeNachDatumAnfrage" />
      <xs:element type="typen:NachrichtenListeNachIDAnfrageTyp" name="
          nachrichtenListeNachIDAnfrage" />
      <xs:element type="typen:ServiceExceptionType" name="serviceException" />

    </xs:schema>
  </wsdl:types>

  <!-- Receive Message -->
  <wsdl:message name="nachrichtHerunterladenAnfrage">
    <wsdl:part element="tns:nachrichtHerunterladenAnfrage" name="
        nachrichtHerunterladenAnfrage" />
  </wsdl:message>
  <wsdl:message name="nachrichtHerunterladenAntwort">
    <wsdl:part element="tns:nachrichtHerunterladenAntwort" name="
        nachrichtHerunterladenAntwort">
    </wsdl:part>
  </wsdl:message>

  <!-- Attachment operations -->
  <wsdl:message name="anhangHerunterladenAntwort">
    <wsdl:part element="tns:anhangHerunterladenAntwort" name="
        anhangHerunterladenAntwort"/>
  </wsdl:message>
  <wsdl:message name="anhangHerunterladenAnfrage">
    <wsdl:part element="tns:anhangHerunterladenAnfrage" name="
```

```
        anhangHerunterladenAnfrage"/>
40  </wsdl:message>
41
42
43  <!-- Message operations -->
44  <wsdl:message name="nachrichtenListeAntwort">
45    <wsdl:part element="tns:nachrichtenListeAntwort" name="
        nachrichtenListeAntwort" />
46  </wsdl:message>
47  <wsdl:message name="nachrichtenListeNachDatumAnfrage">
48    <wsdl:part element="tns:nachrichtenListeNachDatumAnfrage" name="
        nachrichtenListeNachDatumAnfrage" />
49  </wsdl:message>
50  <wsdl:message name="nachrichtenListeNachIDAnfrage">
51    <wsdl:part element="tns:nachrichtenListeNachIDAnfrage" name="
        nachrichtenListeNachIDAnfrage" />
52  </wsdl:message>
53
54  <wsdl:message name="ServiceException">
55    <wsdl:part element="tns:serviceException" name="ServiceException" />
56  </wsdl:message>
57
58  <wsdl:portType name="EmpfangeServicesPort">
59    <wsdl:operation name="nachrichtenListeNachID">
60      <wsdl:input message="tns:nachrichtenListeNachIDAnfrage" name="
          nachrichtenListeNachIDAnfrage" />
61      <wsdl:output message="tns:nachrichtenListeAntwort" name="
          nachrichtenListeAntwortNachID" />
62      <wsdl:fault message="tns:ServiceException" name="ServiceException" />
63    </wsdl:operation>
64    <wsdl:operation name="nachrichtenListeNachDatum">
65      <wsdl:input message="tns:nachrichtenListeNachDatumAnfrage" name="
          nachrichtenListeNachDatumAnfrage" />
66      <wsdl:output message="tns:nachrichtenListeAntwort" name="
          nachrichtenListeAntwortNachDatum" />
67      <wsdl:fault message="tns:ServiceException" name="ServiceException" />
68    </wsdl:operation>
69
70    <wsdl:operation name="nachrichtHerunterladen">
71      <wsdl:input message="tns:nachrichtHerunterladenAnfrage" name="
          nachrichtHerunterladenAnfrage" />
72      <wsdl:output message="tns:nachrichtHerunterladenAntwort" name="
          nachrichtHerunterladenAntwort" />
73      <wsdl:fault message="tns:ServiceException" name="ServiceException" />
74    </wsdl:operation>
75
76    <wsdl:operation name="anhangHerunterladen">
77      <wsdl:input message="tns:anhangHerunterladenAnfrage" name="
          anhangHerunterladenAnfrage" />
78      <wsdl:output message="tns:anhangHerunterladenAntwort" name="
          anhangHerunterladenAntwort" />
79      <wsdl:fault message="tns:ServiceException" name="ServiceException" />
```

```
80      </wsdl:operation>
81    </wsdl:portType>
82
83
84    <wsdl:binding name="EmpfangeServiceSoapBinding" type="tns:EmpfangeServicesPort
          ">
85      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
            http"/>
86      <wsdl:operation name="nachrichtenListeNachDatum">
87        <soap:operation soapAction="" style="document"/>
88        <wsdl:input name="nachrichtenListeNachDatumAnfrage">
89          <soap:body use="literal"/>
90        </wsdl:input>
91        <wsdl:output name="nachrichtenListeAntwortNachDatum">
92          <soap:body use="literal"/>
93        </wsdl:output>
94        <wsdl:fault name="ServiceException">
95          <soap:fault name="ServiceException" use="literal"/>
96        </wsdl:fault>
97      </wsdl:operation>
98      <wsdl:operation name="nachrichtenListeNachID">
99        <soap:operation soapAction="" style="document"/>
100       <wsdl:input name="nachrichtenListeNachIDAnfrage">
101         <soap:body use="literal"/>
102       </wsdl:input>
103       <wsdl:output name="nachrichtenListeAntwortNachID">
104         <soap:body use="literal"/>
105       </wsdl:output>
106       <wsdl:fault name="ServiceException">
107         <soap:fault name="ServiceException" use="literal"/>
108       </wsdl:fault>
109     </wsdl:operation>
110     <wsdl:operation name="nachrichtHerunterladen">
111       <soap:operation soapAction="" style="document"/>
112       <wsdl:input name="nachrichtHerunterladenAnfrage">
113         <soap:body use="literal"/>
114       </wsdl:input>
115       <wsdl:output name="nachrichtHerunterladenAntwort">
116         <soap:body use="literal"/>
117       </wsdl:output>
118       <wsdl:fault name="ServiceException">
119         <soap:fault name="ServiceException" use="literal"/>
120       </wsdl:fault>
121     </wsdl:operation>
122     <wsdl:operation name="anhangHerunterladen">
123       <soap:operation soapAction="" style="document"/>
124       <wsdl:input name="anhangHerunterladenAnfrage">
125         <soap:body use="literal"/>
126       </wsdl:input>
127       <wsdl:output name="anhangHerunterladenAntwort">
128         <soap:body use="literal"/>
129       </wsdl:output>
```

```
130       <wsdl:fault name="ServiceException">
131         <soap:fault name="ServiceException" use="literal"/>
132       </wsdl:fault>
133     </wsdl:operation>
134   </wsdl:binding>
135
136   <wsdl:service name="EmpfangeService">
137     <wsdl:port binding="tns:EmpfangeServiceSoapBinding" name="EmpfangeServicePort
          ">
138       <soap:address location="TO BE DEFINED"/>
139     </wsdl:port>
140   </wsdl:service>
141 </wsdl:definitions>
```

Quelltext B.5: EmpfangeService.wsdl

Literaturverzeichnis

[Eura] EUROPÄISCHE UNION ; EUROPÄISCHE UNION, 1998-2016 http://eur-lex.europa.eu/ (Hrsg.):
Durchführungsverordnung (EG) Nr. 987/2009.
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2009R0987:20130108:DE:HTML. –
Accessed: 2016-06-25

[Eurb] EUROPÄISCHE UNION ; EUROPÄISCHE UNION, 1998-2016 http://eur-lex.europa.eu/ (Hrsg.):
Verordnung (EG) Nr. 883/2004.
http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:02004R0883-20130108. –
Accessed: 2016-06-25

[Gmb] GMBH microTOOL ; GMBH microTOOL (Hrsg.):
Industry 4.0 and the System Context.
http://www.microtool.de/en/requirementsengineering/industry-4-0-and-the-system-context/. –
Accessed: 2016-06-15

[Gra14] GRANDE, Marcus ; VIEWEG, Springer (Hrsg.):
100 Minuten für Anforderungmanagement.
2. Auflage.
Springer Science + Business Media, 2014.
http://dx.doi.org/10.1007/978-3-658-06435-8.
http://dx.doi.org/10.1007/978-3-658-06435-8. –
ISBN 978–3–658–06434–1

[Kle13] KLEUKER, Stephan ; VIEWEG, Springer (Hrsg.):
Grundkurs Software-Engineering mit UML.
3., korrigierte und erweiterte Auflage.
Springer Science + Business Media, 2013.
http://dx.doi.org/10.1007/978-3-658-00642-6.
http://dx.doi.org/10.1007/978-3-658-00642-6. –
ISBN 978–3–658–00641–9

[Nie05] NIEBISCH, Thomas ; VIEWEG, Springer (Hrsg.):
Anforderungsmanagement in sieben Tagen.
2. Auflage.
Springer Science + Business Media, 2005.
http://dx.doi.org/10.1007/978-3-642-34857-0.
http://dx.doi.org/10.1007/978-3-642-34857-0. –
ISBN 978–3–642–34856–3

[PK] PINEDA, Manuel M. ; KLEVENZ, Tobias ; GMBH data2type (Hrsg.):

XSL-FO Einführung in die Sprache für Seitengestaltung und Umbruch.
https://www.data2type.de/xml-xslt-xslfo/xsl-fo/

[PR15] POHL, Klaus ; RUPP, Chris:
Basiswissen Requirements Engineering.
4. Auflage.
dpinkt.verlag, 2015. –
ISBN 978–3864902833

[Rum11] RUMPE, Bernhard ; VIEWEG, Springer (Hrsg.):
Modellierung mit UML.
2. Auflage.
Springer Berlin Heidelberg, 2011.
http://dx.doi.org/10.1007/978-3-642-22413-3.
http://dx.doi.org/10.1007/978-3-642-22413-3. –
ISBN 978–3–642–22412–6

[Sch] SCHEIBL, Hans-Jürgen:
Einführung in die Unified Modeling Language (UML).
http://home.f1.htw-berlin.de/scheibl/uml/index.htm. –
Accessed: 2016-06-04

[SG05] SEEMANN, Jochen ; GUDENBERG, Jürgen W. ; VIEWEG, Springer (Hrsg.):
Software Entwurf mit UML2.
2. Auflage.
Springer-Verlag Berlin Heidelberg GmbH, 2005. –
ISBN 978–3–540–30949–9

[VHH+04] VERSTEEGEN, Gerhard ; HESSELER, Alexander ; HOOD, Colin ; MISSLING, Christian ; STÜCKA, Renate ; VIEWEG, Springer (Hrsg.):
Anforderungsmanagement.
Springer Berlin Heidelberg, 2004.
http://dx.doi.org/10.1007/978-3-642-18975-3.
http://dx.doi.org/10.1007/978-3-642-18975-3. –
ISBN 978–3–642–62388–2

[VS] VOIGT, Kai-Ingo ; SCHEWE, Gerhard ; VERLAG, Springer G. (Hrsg.):
Gabler Wirtschaftslexikon.
http://wirtschaftslexikon.gabler.de/Archiv/13507/projekt-v7.html. –
Accessed: 2016-06-02

[w3s] W3SCHOOLS.COM ; DATA, Refsnes (Hrsg.):
XSLT Tutorial.
http://www.w3schools.com/xsl/default.asp

[WAa] WIKIPEDIA-AUTOREN ; WIKIPEDIA, Die freie E. (Hrsg.):
File:UML-Diagrammhierarchie.svg - Wikimedia Commons.
https://commons.wikimedia.org/wiki/File:UML-Diagrammhierarchie.svg. –
Accessed: 2016-06-15

[WAb] WIKIPEDIA-AUTOREN ; WIKIPEDIA, Die freie E. (Hrsg.):
Profil (UML).
https://en.wikipedia.org/wiki/Profile_(UML). –
Accessed: 2016-06-04

[WAc] WIKIPEDIA-AUTOREN ; WIKIPEDIA, Die freie E. (Hrsg.):
Projekt.
https://de.wikipedia.org/wiki/Projekt. –
Accessed: 2016-06-02

[WAd] WIKIPEDIA-AUTOREN ; WIKIPEDIA, Die freie E. (Hrsg.):
Single Point of Contact.
https://de.wikipedia.org/wiki/Single_Point_of_Contact. –
Accessed: 2016-07-01

[WAe] WIKIPEDIA-AUTOREN ; WIKIPEDIA, Die freie E. (Hrsg.):
Sozialverischerung (Österreich).
https://de.wikipedia.org/wiki/Sozialversicherung_(%C3%96sterreich). –
Accessed: 2016-07-10

Glossar

Apache Maven Apache Maven
Buildtool zum automatisierten Erzeugen von Artefakten und zur Abhängigkeitsauflösung
Site:`http://maven.apache.org`

Artefakt Als Artefakt bezeichnet man alle Elemente, welche während des Erstellens der Softwarekomponenten erzeugt werden, z.B. eine JAR - Datei

CDATA Character DATA ist ein Abschnitt innerhalb eines XML-Dokumentes, welches Teil des Dokuments ist, aber nicht interpretiert werden soll. Hier werden z.B. binäre Daten mittels Base64 codiert übergeben, aber auch XML-Fragmente, welche nicht innerhalb des XSD definiert wurden.

CXF CXF ist der Zusammenschluss aus den Frameworks *IONA Celtix* und *Codehaus XFire* und dient der Implementierung von Server- und Client-Elementen im Bereich der Webservices

EA EnterpriseArchitect ist ein Modellierungstool der Firma Sparx Systems für UML. Weiter Information findet man unter `http://www.sparxsystems.at`

EJB EnterpriseJavaBeans dienen der Kommunikation zwischen einem Server und einem Client(Remote) und können auch zur einfacheren Kommunikation innerhalb eines Applikationsservers eingesetzt werden(lokal)

FOP Apache FOP ist ein FO-Prozessor, welcher auf JAVA basiert und aus einer FO-Datei ein PDF, oder auch ein PostScript machen kann. Nähere Informationen dazu findet man unter `https://xmlgraphics.apache.org/fop/`

JAXB JAXB ist der Ansatz, dass aus XML-Dateien mittels JAXB Javaklassen befüllt werden. Das Binding bewirkt hierbei, dass aus den einzelnen XML-Datentypen JAVA-Datentypen werden. Weitere Informationen zum Thema JAXB findet man unter `https://jaxb.java.net`

Oracle Oracle Corporation
500 Oracle Parkway, M/S 5op7
Redwood Shores, CA 94065
Site:`http://www.oracle.com`

PDF Das Portable Document Format kurz PDF ist ein plattformunabhängiges Dateiformat für Dokumente, das vom Unternehmen Adobe Systems entwickelt und 1993 veröffentlicht wurde und das Ziel verfolgt, dass der Ausdruck immer so aussieht, wie es geschrieben wurde

RE Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen

RedHat Red Hat, Inc. ist eine Softwarefirma mit Sitz in Amerika.

SPoC Als Single Point of Contact (SPoC) wird in einer Organisation eine zentrale (also einzig mögliche) Anlaufstelle für ein bestimmtes Thema oder Problem oder eine festgelegte Tätigkeit bezeichnet [WAd]

Stakeholder Der Stakeholder ist eine zentrale Rolle im Requirements Engineering (RE). Er dient der Wissensgewinnung, was das System leisten soll und in welchen Grenzen es sich bewegt

SysML SysML ist eine Modelierungssprache, welche im Bereich des System Engineerings zur Anwendung kommt. SysML wurde im Jahr 2003 von der OMG aufgenommen und weiterentwickelt

UML Unified Modeling Language, eine standardisierte Beschreibungssprache für Strukturen und Abläufe in objektorientierten Programmsystemen

UUID Ein Universally unique identifier (UUID) ist ein Standard für Identifikatoren, der in der Softwareentwicklung verwendet wird und aus einer 16-Byte großen Zahl besteht. Die Form der UUID entspricht folgenden Pattern:
$[0-9a-fA-F]\{8\}-[0-9a-fA-F]\{4\}-[0-9a-fA-F]\{4\}-[0-9a-fA-F]\{4\}-[0-9a-fA-F]\{12\}$

Versicherungsträger Ein österreichischer Sozialversicherungsträger ist eine Einrichtung, welche dem Hauptverband der österreichischen Sozialversicherungsträger angehört. Vergleiche [WAe]

XPath XPath ist ein Such- und Auswahlsprache zum Suchen und Selektieren von Knoten in XML-Dateien. Weitere Informationen und ein Tutorial findet man unter `http://www.w3schools.com/xsl/xpath_intro.asp`

XSL eXtensible Stylesheet Language ist eine erweiterte Layoutierungssprache zum transformieren von XML in ein anderes Format, z.B. HTML, oder in ein anderes XML-Format

XSL-FO eXtensible Stylesheet Language - Formatting Objects ist die Erweiterung von XSL, um daraus eine FO - Datei zu erzeugen, welche dann in ein Ausgabeformat, meist PDF, konvertiert wird.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 23. September 2016